Tatouage des bases de données

David Gross-Amblard

Université de Bourgogne
École doctorale E2S

Habilitation à diriger des recherches
Discipline Informatique

David Gross-Amblard

Tatouage des bases de données
Databases watermarking

Soutenue le 6 décembre 2010, à Dijon

Rapporteurs / Referees

M. Alban Gabillon	Université de la Polynésie Française
M. Sanjeev Khanna	University of Pennsylvania
M. Dan Suciu	University of Washington

Examinateurs / Examiners

M. Serge Abiteboul	INRIA-Saclay et Académie des Sciences
M. Michel Scholl	CNAM-Paris
M. Kokou Yetongnon	Université de Bourgogne

version 1.0

Contents

Acknowledgements / Remerciements

First of all, I would like to thank my referees, Alban Gabillon, Sanjeev Khanna and Dan Suciu, for making me the honor of being part of my jury. Dan, thank you for the numerous comments I obtained on these works. Sanjeev, I would like to thank you for your inspiring SODA paper: you can see that I really enjoyed it. Alban, pour avoir accepté de lire l'HDR de l'encadrant et la thèse de l'encadré, et pour m'avoir honoré de ta présence en métropole, passant ainsi d'un climat tropical à un climat océanique à tendance semi-continentale (bref, il fait froid à Dijon et beau à Pape'ete).

J'aimerais ensuite remercier mes examinateurs, Serge Abiteboul, Michel Scholl, et Kokou Yetongnon. Vous êtes ou avez été chacun mes employeurs, je respecterai donc l'ordre chronologique d'embauche.

Michel : je te remercie sincèrement d'avoir parié sur moi. C'est grâce à toi que je peux faire ce métier. L'ambiance de l'équipe Vertigo doit beaucoup à ta façon d'être.

Kokou : je te suis reconnaissant de m'avoir accueilli dans l'équipe SI&SI, alors que j'étais "loosely coupled". Maintenant, c'est "strongly".

Serge : je me rappelle de ton hospitalité et de ton humour alors que j'étais un jeune doctorant à sa première conférence, Dallas, Texas (2000). Dix ans plus tard, tes qualités sont inchangées, seul le nom du projet est différent.

Toujours dans le même ordre, je voudrais remercier les membres des différentes équipes que j'ai fréquentées en 9 ans : Vertigo, SI&SI et WebDam. Tout d'abord l'équipe Vertigo : Bernd Amann, Nouha Bouteldjia, Camélia Constantin, Vassilis Christophides, Michel Crucianu, Cédric Du Mouza, Irini Fundulaki (from NY to Edinbourgh), Valérie Gouet-Brunet, Julien Lafaye (on s'est bien marré non ?), Radu Pop, Sébastien Poullot, Imen Sebei, et Dan Vodislav.

En passant, j'aimerais saluer Marie-Christine Costa, directrice du laboratoire Cedric pour son soutien. Je remercie également Bernard Lemaire, Agnès Plateau et F.-Y. Villemin pour les bons moments d'enseignement au CNAM de Paris.

Je remercie également les membres de l'équipe SI&SI : Elie Abi-Lahoud, Lylia Abrouk, Bechara Al Bouna, Richard Chbeir, Christophe Cruz, Nadine Cullot, Elisabeth Gavignet, Raji Ghawi, Damien Leprovost ("so say we all"), Christophe Nicolle, Elie Raad, Sylvain Rampacek (ré !), Mônica Ribeiro Porto Ferreira, Fekade Getahun Taddesse, et Joe Tekli, dans l'attente de connaître les nouveaux arrivants. Je n'oublie pas mes camarades de classe de l'Université de Bourgogne : Jean-Luc Baril, Joël Savelli et Olivier Togni (proof-checkers).

Richard, un mention particulière pour ta générosité. Et si on l'écrivait ce second papier ?

Enfin, je remercie mes nouveaux collègues du projet WebDam : Emilien Antoine, Meghyn Bienvenu, Alban Galland, Bruno Marnette, et Marie-Christine Rousset. Dans et autour de WebDam : Pierre Bourhis, Ioana Manolescu, Luc Segoufin et Victor Vianu.

Pour Luc : le calcaire c'est fait. À quand le granite ?

Il y a les équipes, mais aussi les projets de recherche. Pour le tatouage, les membres du projet Tadorne : Cristina Bazgan, Cyril Bazin, Jean Béguec, Meryem Guerrouani, Jean-Marie Le Bars, Jacques Madelaine, Ammar Mechouche, et Anne Ruas. Je remercie également les autres tatoueurs de bases de données, Jerry Kiernan, et Radu Sion, et les tatoueurs en général, Caroline Fontaine et Teddy Furon. Jan Van den Bussche and Michael Benedikt for their precious remarks on early versions of this work (Jan, you were right, submitting was a good idea).

Cristina, merci pour ton aide et tes conseils. Je compte sur toi pour réduire mon nombre d'Erdös.

Place à la musique. Je salue les membres du projet NEUMA : Hervé Audéon, Cécile Davy-Rigaux, Zoé Faget, Philippe Rigaux, Alice Tacaille et Virginie Thion-Goasdoué.

Je remercie tout particulièrement et très chaleureusement Philippe, Cécile, ainsi que Clément et Simon, pour leur accueil du *visiteur du jeudi* et pour leur Tempérance.

Je salue enfin mes compères du Cedric, Pierre Courtieu et Xavier Urbain. Des vrais potes quoi. Toujours prêts pour un café ou une andouillette grillée. Xavier, hôtellerie cinq toques et cinq clés.

Les week-ends à Dijon ne seront plus pareils sans Nicolas Roussel, Rosane Ushirobira, et leur galopins : Thaïs, Maxime et Sophia. Mais la France, c'est petit ! Un clin d'oeil à Werner Seiler (un jour nous auront notre réponse), à Fabienne Hébrard (miam) et à Philippe Pagniez (l'informatique du futur).

Il y a aussi Marcel et Margot, Nicole et Pierre, Bernard, Vincent, Yvette et André : faut-il dire quelque chose ?

Enfin, je remercie Géraldine pour son soutien permanent. Si son amour est aveugle à mes défauts, il n'en est pas moins robuste aux agressions du temps. Et faut dire qu'elle a comme ... un p'tit chat sauvage dans les yeux, qui ressemble au tatouage que j'ai dans l'coeur. Y'a pas d'erreur.

Figure 1: Un nuage

Avant-propos (French)

Dans ce document je présente une vue d'ensemble de mes travaux concernant le tatouage de bases de données, coeur de mon activité depuis 2001. Mes autres centres d'intérêt sont présentés en annexe A.

En bref

Depuis septembre 2006 je suis membre de l'équipe Systèmes d'information et systèmes d'images (SISI) du laboratoire Le2i de l'université de Bourgogne, équipe dirigée par Kokou Yetongnon. Je suis actuellement en délégation INRIA dans le projet ERC Grant WebDam de Serge Abiteboul. Précédemment, j'étais membre de l'équipe Vertigo - Bases de données du laboratoire Cedric (Cnam-Paris) depuis septembre 2001, alors sous la direction de Michel Scholl. J'ai effectué une thèse de doctorat sous la direction de Michel de Rougemont dans l'équipe algorithmique et complexité du LRI (Paris XI), alors dirigée par Miklos Santha.

Après avoir travaillé sur l'*approximation de requêtes* dans les bases de données géographiques (publication dans la conférence ACM PODS [38] et la revue JCSS [41]), je travaille actuellement à la *sécurisation des bases de données par tatouage*. J'ai initié cette activité dans l'équipe Vertigo (une thèse soutenue, publication dans les revues IEEE TKDE [67] et ACM TODS [40], et dans la conférence ACM PODS [39], réalisation d'un logiciel [27], coordinateur d'une ACI sur ce thème [4]), et la poursuit actuellement au sein de l'équipe SISI dans le cadre de l'ANR NEUMA [6] (conférence ISMIR 2009 [43]). Je m'intéresse également au classement de services Web à la Google (une thèse soutenue, publication dans la Revue des sciences et théorie de l'information [25] et la conférence OTM COOPIS [24]) et à la publication de données sur le Web (publication dans la Revue des sciences et théorie de l'information [45] et la conférence ICWE [46]).

Déroulement de carrière

2010-2011 Délégation INRIA dans le projet ERC Grant WebDam de Serge Abiteboul.

2006- Maître de conférence titulaire, Université de Bourgogne.

2001-2006 Maître de conférence titulaire, Conservatoire national des arts & métiers, Paris.

2000-2001 ATER, Orsay, Université Parix XI.

1996-2000 Doctorat en Sciences, spécialité Informatique, intitulé "Approximation dans les bases de données contraintes", Orsay, université Paris XI, décembre 2000.

1996-2000 Moniteur rattaché à l'université Paris XI, durant les trois années de la thèse.

1998-1999 Scientifique du contingent, Laboratoire d'Informatique de Polytechnique (LIX).

1995-1996 D.E.A. d'Informatique, Orsay, université Paris XI.

1993-1995 Licence et maîtrise d'Informatique, Orsay, université Paris XI.

Alice: *Look. I am old enough now. I think I can bear a pretty tatoo on my shoulder.*
Maestro: *Thanks for using the french vocable, but I think that you mean a "watermark".*
Roberto: *At least a watermark will be invisible.*
Augusto: *And robust ! Just remember to keep a small part of it secret.*
Alice: *Ok then, let's go for a watermark.*
Eva: *But if you do so, you will have to respect some constraints...*

1

Introduction (English)

This introductory chapter gives a brief overview of watermarking and database watermarking specificities. It presents an informal survey of the main results.

Digital watermarking

Informally, digital watermarking is a voluntary alteration of an electronic document, in order to attach a message – a watermark – to it. Applications of watermarking are numerous, including:

- Intellectual property protection: in various scenarios, a data owner/provider has spent time and efforts to build high quality documents (for example terrain explorations to devise an accurate geolocalized data set). But due to the digital nature of these data, the legitimate owner is threatened by unfair customers, reselling illegal, perfect copies of the document. By hiding the owner's identity into a document, watermarking offers the ability to prove ownership once a suspect document has been found (Figure 2.1). A natural example is the Digimarc watermarking plugin for Adobe Photoshop[1].
- Fingerprinting: instead of hiding only the owner's identity, hiding the customer's identity into the document allows to track back the exact malevolent customer reselling copies (Figure 2.2). A classical example is the detection of Academy award voters that helped illegal broadcast of so-called screener copies of films[2].
- Meta-data hiding: dissimulating in a document its unique id number, or the exact technical parameters used for data acquisition, guarantees that these meta-data will remain permanently attached to the document, whatever format transformations or file manipulation occurring in the future.

Classically, a watermarking protocol uses two algorithms, the *marker* and the *detector*, that respectively hide and extract a *watermark*. The watermark is usually thought as being invisible, hence watermarking shares some similarities with information hiding techniques (that rather deal with secret communications, disregarding the very nature of the document used). Most applications consider also watermarks that are robust to malevolent operations from attackers wishing to erase them (there exists also visible and/or fragile watermarking techniques, but we do not consider them in this work). Common constraints on watermarking systems are:

- Invisibility: hiding a watermark should not impact the intended use of the document (should not lower its quality beyond a reasonable limit). It is noteworthy that a watermarking method is doomed to alter the original document to be efficient. Indeed, hiding information in the document representation (in unused bits for example) is extremely sensitive to informed attackers. Hence, watermarking has to alter the data semantics, but in a restricted way.

[1] `https://www.digimarc.com/solutions/`
[2] `http://www.msnbc.msn.com/id/4037016/`

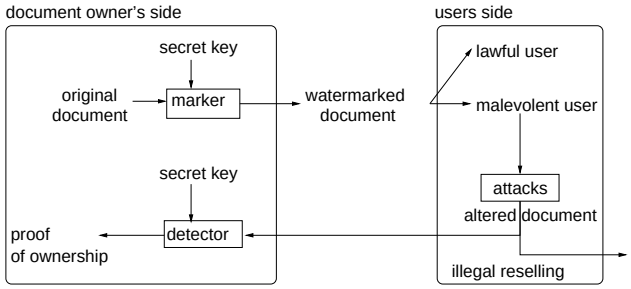


Figure 1.1: Watermarking scenario

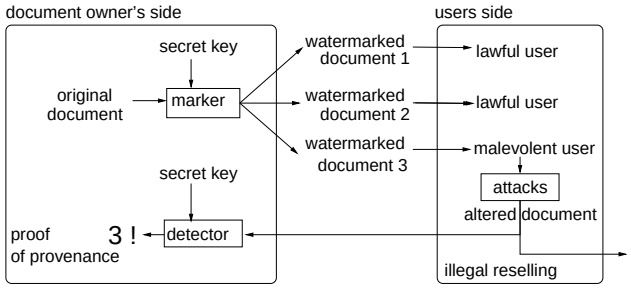


Figure 1.2: Fingerprinting scenario

- Controlled capacity: the amount of data that can be hidden in a document should be predictable. A high capacity if often searched for.
- Low false-positive rate: the probability to detect a message in a non-watermarked document must be negligible.
- Robustness: the detector should be able to detect the hidden message in reasonnably altered data sets by a malevolent attacker. The attacker is nevertheless limited as he needs a still salable document.
- Public access to the algorithm: the security of the method should not rely on the secrecy of the algorithm, but on a private secret key only (Kerckhoffs'principle [59, 60]).
- Security: an attacker should not infer the watermark localization or content, or the secret key [20].
- Blindness: the detector should operate without the use of the original, unwatermarked document.

There is a natural trade-off between the watermark robustness and its invisibility: a more robust watermark requires a stronger alteration of the original document. The attacker is also limited by the invisibility constraint: the attack (alteration) of the watermark document must be limited, so that the attacked document remains valuable.

Database watermarking

While original watermarking techniques mainly arose in the multimedia domain for images, sound or video, they have natural applications in databases. The amount of structured data available on the Internet is drastically increasing, with source ranging from public agencies (e.g. environmental measurements) to professional data sets (sales databases, stock exchange databases, customers profiles, etc.). There exist also examples from the past where structured data sets have been modified for intellectual property protection.

A famous specimen is the use of contrived number rounding conventions by editors of onerous logarithm tables (Figure 2.3).

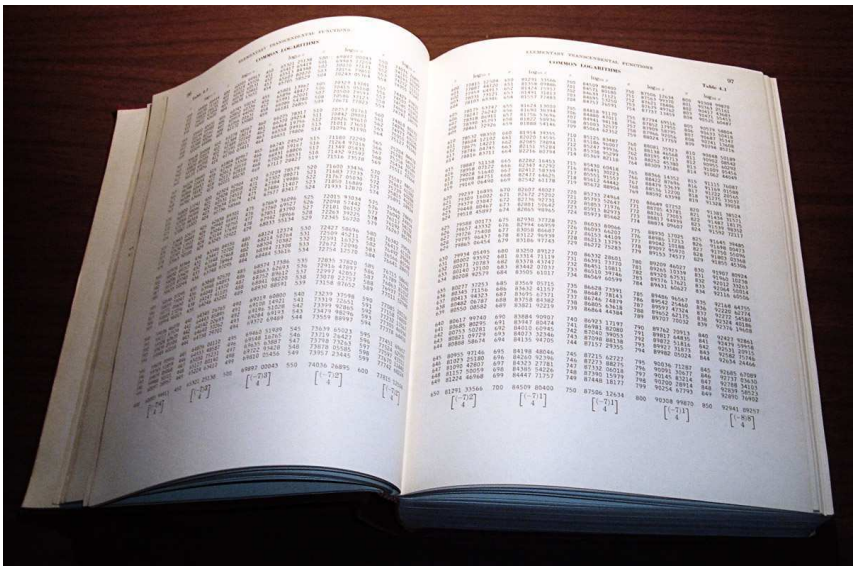


Figure 1.3: Paper logarithm table: number rounding conventions where used for ownership proofs (source:Wikipedia)

Modern data sets represented as database show interesting specificities that lead to new questions for digital watermarking:

- Interwoven relationships between data: while an image is a regular juxtaposition of pixels, a data set describes various irregular relationships between tuples, without a clear ordering between them. This yields a synchronization problem to the marker and detector, that require to precisely locate the watermark in the data.
- Shared semantics: a data set is usually used among other sources, for example joining hotels data with touristic roadmaps. Hence there is a shared semantics between data sets, for a sound naming of objects (e.g. name of a road in both hotel and roadmap data sets). This limits the attacker possibilities, as an altered data set still have to comply with this shared semantics to be valuable.
- Multiple data types: numbers, categorical data, streams, spatial data, etc.
- Versatile use: oppositely to multimedia documents, that are used in a limited number of ways, a data set can be explored in various ways, through expressive query languages.
- Many-faceted quality constraints: usually, multimedia quality is expressed as a global measure between the original and watermarked document, or between watermarked and attacked document (like peak-signal to noise ratio – PSNR) . On the contrary, a database is prone to formal quality constraints: tuple-wise data accuracy, functional dependencies between tuples and in a general setting, any application-defined constraint expressed in a general query language (e.g. semantic integrity constraints).
- Access model: data sets can be accessed by the detector as plain tables, but also only through specific views.
- Incrementality: in many contexts, a data set is a temporal object, that needs regular updates to remain accurate (for example, a variation of 10% can be observed between two updates of national geographical databases [91]). Hence there is a need to maintain the watermark as long as updates are propagated to legitimate customers.

- Efficiency: because databases are usually huge and part of a general data management system, the watermarking procedure has to be included into the system and hence should perform efficiently, according to databases systems standards. This is relevant specifically for data sets with a short-time value, like e.g. weather forecasts, stock exchange real time values and so on.

The following example presents a database watermarking scenario, where database-specific quality constraints are illustrated.

Example 1 *Tables below present a touristic database instance. For this application, the goal is to hide information by slightly modifying transport prices, under the following quality constraints devised by the data owner:*

- C_1*: Allow a distortion* ± 10 *of each price (tuple-wise, accuracy constraint).*
- C_2*: Allow is distortion* ± 20 *on the total of all prices (relation-wise aggregate constraint, without parameter).*
- C_3*: For* any *travel* t*, a distortion* ± 10 *is allowed on the total price of travel* t *(relation-wise,* parametric *aggregate constraint).*

Route:

travel	*transport*
India discovery	*T1*
India discovery	*T2*
Nepal Trek	*T1*
Nepal Trek	*T3*
Nepal Trek	*T4*
TourNepal	*T4*
TourNepal	*T5*

PriceTable:					*PriceTable′:*	*PriceTable″:*
transport	*departure*	*arrival*	*type*	*price*	*... price*	*... price*
T1	*Paris*	*Delhi*	*plane*	*35*	*45*	*25*
T2	*Delhi*	*Nawal.*	*bus*	*20*	*30*	*30*
T3	*Delhi*	*Kathm.*	*plane*	*15*	*25*	*5*
T4	*Kathm.*	*Simikot*	*plane*	*30*	*20*	*40*
T5	*Kathm.*	*Daman*	*jeep*	*50*	*10*	*40*
T6	*Kathm.*	*Paris*	*plane*	*10*	*10*	*10*

PriceTable represents the original instance of the data owner. PriceTable′ and PriceTable″ are two watermarked instances (prices are modified). PriceTable′ breaks constraint C_3 *, because the cost of the India discovery travel is now 75 instead of 55. PriceTable″ respects all the former quality constraints.*

According to the previous example, the search of valid watermarks can turn into a difficult combinatorial problem, for general quality constraints.

Line of research

Various initial propositions on database watermarking where announced independently at VLDB'2002 by Rakesh Agrawal and Jerry Kiernan, SIGMOD'2003 by Radu Sion, Mickael Attalah and Sushil Prabakhar, and PODS'2003 for our work. On the one hand, Agrawal and Kiernan proposed a complete, blind watermarking method for numerical databases, as part of their Hippocratic Databases project [9]. They did not consider the

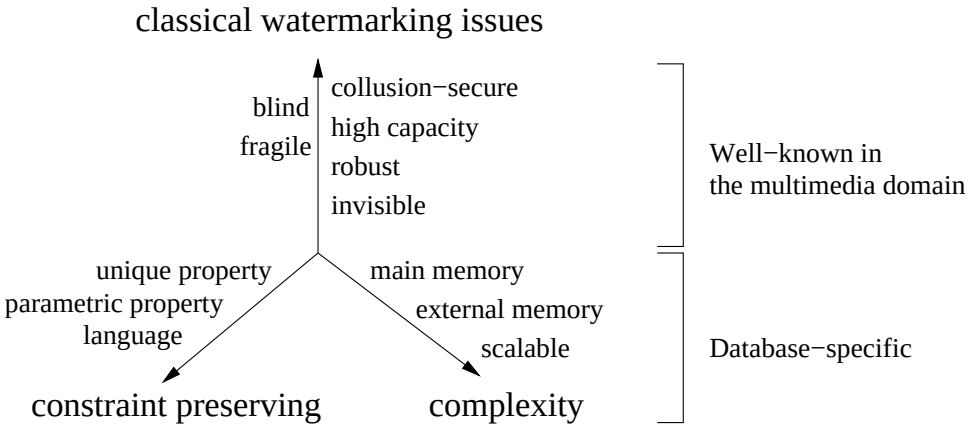


Figure 1.4: Criteria for database watermarking

preservation of the result of queries, but they observed that the mean and variance of numerical attributed are slightly altered by watermarking. On the other hand, Radu Sion et al. and myself proposed the first query-preserving watermarking methods[3]. Radu Sion et al. technique was oriented toward a practical solution, and the search for query-preserving watermarks was performed through a greedy search method.

Our work was started at Cedric Lab., CNAM-Paris, in Michel Scholl's Vertigo team (now directed by Michel Crucianu). This study first explored these issues on the theoretical side, following the methodology developed in the database field, with a new security flavor:

- Expressing the database watermarking problem by logical means: expressing the quality constraints a watermarking procedure must respect using a logical or practical language.
- Describing the set of valid watermarks that respects all quality constraints, according to syntactical and structural properties of constraints and data sets. This means also providing lower bounds on the watermarking capacity of data sets, or impossibility results. As a side effect, this yields lower and upper bounds on the unavoidable alterations of data.
- Obtaining a practical solutions with a simple user-oriented constraint language, with optimizations at the logical and physical level.
- Generalizing to specific data types manipulated by databases.
- And of course, complying with all concerns of any watermarking system (invisibility, high capacity, robustness, blindness, etc.)

This line of research addresses some specific aspects of database watermarking, compared to the classical work in the multimedia domain. These specificities are summed up on Figure 2.4.

In the sequel, we present a global overview of our results: the theoretical study of query-preserving watermarking in Chapter 3, the obtention of efficient algorithms in Chapter 4, and variations around these techniques for several application domains in Chapters 5, 6 and 7. The chosen presentation is thematic, and does not respect the exact chronology of papers publication.

Chapter 3: query-preserving watermarking

The first chapter of this work focuses on modelling basic quality constraints for numerical data sets, by means of aggregate sum queries (published in PODS 2003 [39] and TODS 2010 [40]). We consider, borrowing the

[3]A property-preserving method for graphs was nevertheless proposed by Sanjeev Khanna and Francis Zane in 2000 [61]. We discuss this work later on and in depth in Chapter 3.

now classical approach of both database theory and descriptive complexity, the relationship between the watermarking capacity on the one side, and the expressive power of the query language used for constraint specification on the other side. For example, constraint C_3 of Example 1 is equivalent to preserving the sum of prices of transports v for a given travel u, selected by the first-order query

$$\psi(u, v) \equiv \exists x_1 x_2 x_3 x_4 x_5 Route(u, v) \wedge PriceTable(v, x_1, x_2, x_3, x_4, x_5).$$

In this direction, but not in a database perspective, Khanna and Zane [61] proposed a protocol with guaranteed capacity for a specific *parametric* query: shortest path queries on weighted graphs. Their information insertion does not modify the length of shortest path between *any* pair of vertices beyond an acceptable distortion. The watermarking capacity is $\Omega(n^{1/2-1/d})$ for distortion d, for any graph with n edges. From the theoretical point of view, they observe that shortest path queries have a low computational complexity, and suspect that watermarking protocols for NP-hard search spaces are difficult to analyze.

Starting from this approach, we have first generalized their model to databases instances with numerical values (not only weighted graphs). We have also considered the preservation of properties expressed in a query language, not only a specific property like shortest paths.

For a brief overview of the obtained results, the main interesting parameters are the size of the database instance that we denote n in this introductory chapter, and the maximal distortion occurring on the quality constraints, denoted by d. For the sake of simplicity, alteration on each numerical attribute in either +1 or -1, without loss of generality.

First, consider for example the problem of computing the total number of valid watermarked Travel databases of Example 1, i.e. the watermarking capacity of this database. It turns out that the general problem is difficult:

Theorem 2 *(informal) Computing the watermarking capacity is #P-complete.*

Then, we are interested in lower-bounds on this capacity, taking into account the trade-off with data alteration.

Theorem 3 *(informal) For any set of constraints and any database instance, there exists a watermarking scheme with capacity $\Omega(d \log n)$ and error at most d.*

But $d \log n$ is considered as a minute capacity, and one is interested in n^c-capacity watermarking schemes, for a constant c. A first result is that, if the constraints are a finite set of non-parametric queries (like constraint C_2 in example 1), the watermarking capacity is huge:

Theorem 4 *(informal) For any finite set of non-parametric constraints and any database instance, there exists a n-capacity watermarking scheme with constant error 0.*

The main idea is to find two tuples that participate in and impact exactly the same set of constraints, and to modify their values oppositely. Recalling Example 1, tuples $T1$ and $T2$ impact both the travel *India discovery*. Adding 1 to the price of $T1$ and substracting 1 on $T2$ (or the contrary), has no impact on the travel cost. Each of such *compensating pairs* allows to hide one bit of information. If we turn to parametric queries (like constraint C_3), which yields intricate sets of constraints, finding such pairs is harder. Indeed, we first show that the watermarking impact is unbounded if no hypothesis is given. Indeed, using tools from PAC learning theory [113], namely the Vapnik-Chervonenkis dimension [16], the set of tuples is shattered by the parameters : there is always a parameter value that can isolate the very place where the +1 and -1 alterations stand:

Theorem 5 *(informal) There exists a parametric query and a database instance without constant-error n^c-capacity watermarking schemes.*

Then, we use restrictions on data sets that were proposed in another context, i.e. the study of the complexity of structures according to their degree or their tree-width (which measure its similarity with a tree). In this setting, the following results were obtained:

Theorem 6 *(informal) There always exists constant-error n^c-capacity watermarking schemes for:*

- *parametric local queries on structures with bounded-degree Gaifman graph (for example, simple SQL queries on bounded degree graphs). Local languages includes FO (SQL), order-invariant queries or $AGGR_{\mathbb{Q}}$ queries (basic SQL with aggregates).*
- *parametric monadic second order queries on trees or structure of bounded tree width (for example XML trees with small XPath fragments).*

Using tools from Gröhe and Tùran [37], one can show that, without these hypothesis, no watermarking can be obtained. Moreover, in a recent work [40], we have shown that:

Theorem 7 *(informal)*

- *There exists a parametric query of arity k such that any n^c-capacity watermarking scheme has at least error k.*
- *For the previous class of database instances and queries, there exists a corresponding watermarking scheme with error at most k (hence an optimal scheme).*

These results show that, for natural databases, there is a huge watermarking capacity. But our first theoretical solution did not fulfill classical watermarking properties. For example, the algorithm was not blind (the whole original data set is required for detection). Exploring the watermarking space, and obtaining an (almost) blind algorithm is the subject of the second chapter.

Chapter 4: practical aspects, the WATERMILL system

The second chapter focuses on practical aspects of watermarking, to achieve a full system dedicated to database watermarking. This study was done during the ACI Tadorne project[4], funded by the French national research agency (ANR), of which I was the initiator and coordinator. During this project, the design of a simple and user-oriented constraint language was performed. The previous constraints of Example 1 can be expressed by the following declarations:

```
(C1) local 10  on price
(C2) global 20 on (select sum from price)
(C3) forall t in (select route from travel)
          global 10 on (select sum(price) from route where travel=t)
```

The proposed real watermarking algorithm relies on the discovery of compensating pairs, as proposed in the previous chapter. On the database side, the main problem is to reach scalability for their discovery. The used technique is to translate the search of these pairs into a unique SQL query, that is devoted to the RDBMS. On the security side, various improvements where made. First, we adapted an algorithm proposed by Agrawal and Kiernan [7, 8] (that did not take relation-wise quality constraints into account) to obtain watermark synchronization and security. We obtain data-blindness of the algorithm by replacing the simple method of compensating pairs (that requires the original for detection) by opposite bit exchange into values (Example 8).

[4] http://ufrsciencestech.u-bourgogne.fr/~gadavid/tadorne/

Example 8

Absolute value compensation (non-blind)

transport	*price (original)*	*price (watermarked)*
T1	*35*	*25 (-10)*
T2	*45*	*55 (+10)*
total	*80*	*80*

Opposite bit exchange (data-blind)

transport	*price (original)*	*price (watermarked)*
T1	$(35)_2 = 10\boxed{0}011$	$10\boxed{1}011 = 43$
T2	$(45)_2 = 10\boxed{1}101$	$10\boxed{0}101 = 37$
total	*80*	*80*

The method is semi-blind: the whole data set is no longer needed, but the set of watermarked positions has to be memorized and produced at detection time (as in all other existing query-preserving watermarking methods [103]). The algorithm is also equipped with a sophisticated collusion-secure fingerprinting scheme due to Tardòs [109]. The complete method [26, 67] obtained with Julien Lafaye, Camélia Constantin and Meryem Guerrouani, was implemented into the GPL software WATERMILL[5], and validated on huge data sets. It was possible to watermark 1,000,000 tuples respecting 100 constraints in a few minutes (while other methods requires hours or days).

Chapters 5, 6 and 7: Specific algorithms

The next part of our work consisted in the design of database watermarking algorithms for specific datatypes. Here, the basic skeleton of the algorithm is Agrawal and Kiernan's, and the focus is on the very specificities of the application: mainly XML streams, symbolic musical databases, geographical databases and multimedia databases. We focus here on the three first kind.

Chapter 5: Typed XML streams

The first specific method considers the watermarking of XML streams. In the classical streaming context, data has to be processed in a memoryless manner: any operation like type validation or data transformation has to be performed by a finite state automaton (Figure 2.5). In this work with Julien Lafaye [66], we considered the problem of watermarking XML structures while preserving their type, expressed by a non-recursive DTD. It is known that typechecking such streams can be performed by finite automaton [101]. We watermark such streams by copying the stream during its typechecking, but we sometimes perform a *detour* into equivalent runs of the automaton. The performed error is controlled by the edit distance between the original and watermarked stream.

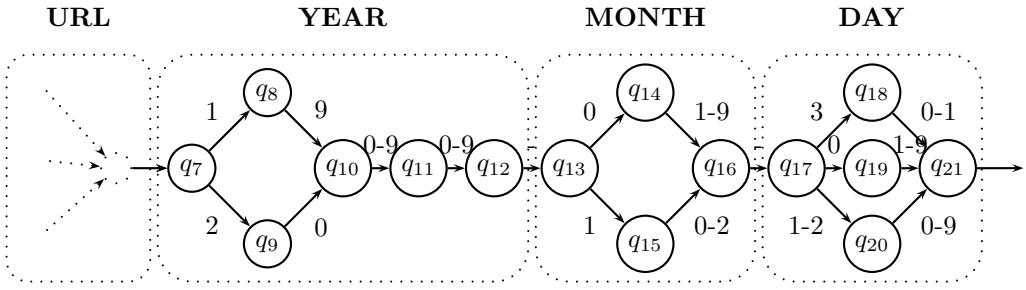


Figure 1.5: A partial specification of a stream type for a `date` element

Chapter 6: Symbolic musical scores

A second result was obtained when I joined the Le2i-CNRS Lab, in the SISI team directed by Kokou Yetongnon. After developping a watermarking method for multimedia data with Richard Chbeir [21], I

[5] `http://watermill.sourceforge.net`

considered music databases. This work was part of the Neuma project[6], also an ANR funding, initiated in 2008 and directed by Philippe Rigaux. Music is considered in a symbolic representation, i.e. not as sound files or music score sheets images, but by the fine descriptions of notes, rhythms and annotations, for example in MusicXML. Part of this system is a set of watermarking solutions for symbolic descriptions. In this approach, we saw musical scores as streams of notes. The first solution is the watermarking of a useful annotation for beginner musicians, fingering annotations. Basically, a fingering is a choice of which finger to use to produce each note. From the computational point of view, the quality of a fingering is a function of the difficulty to play the fingering, and is related to the human hand capabilities (several works model such capabilities). The main idea is to watermark the score by choosing specific fingerings into the space of all possible fingerings. The original fingerings are usually hand-made and of a very high quality. The challenge is then to produce correct fingerings with a controlled alteration (Figure 2.6). We obtained such a method [43] with Philippe Rigaux, Lylia Abrouk and Nadine Cullot.

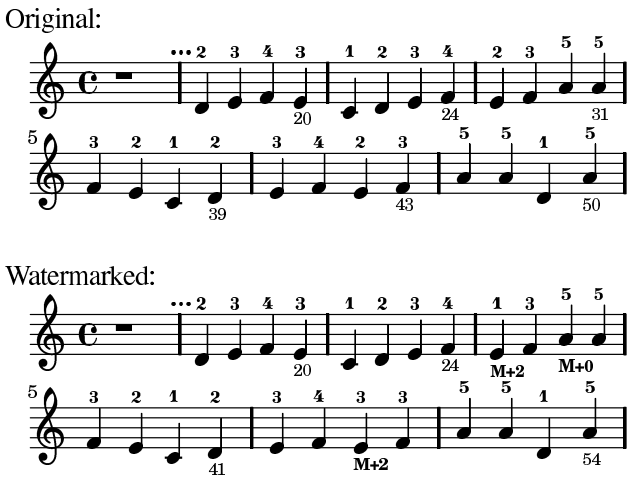


Figure 1.6: An original score with a high-quality fingering, and its watermarked counterpart. Fingering annotations appear upon the staff (right hand, 1: thumb,...5: little finger). Below is indicated the physical cost of playing the fingering. The watermarked version is harder to play, with three altered positions indicated by an M.

Chapter 7: Geographical data

This last result concerns geographical data sets, and was also obtained during the Tadorne project with the Cogit Lab (IGN, National Geographical Institute), along with Julien Lafaye, Jean Béguec and Anne Ruas. We provided a solution for vectorial maps used for their precision, with a focus on the building layer (the biggest part of professional data sets). The main problem with respect to the related work is to obtain robust identifiers for polygons, and to take into account the specific quality metrics use in geographical applications. In this work, we relied on the presence of a common reference system to reason about positions (the WGS 84 GPS system for example). Then we constructed robust identifiers of building by choosing the highest significant bits of the coordinates of their centroid (an attacker has to perform huge transformations in order to alter these bits). Finally, data hiding was performed through scaling of the building according to its main orientation, as shown in Figure 2.7.

This very simple method yields interesting properties, and mainly has a small impact on the angular quality of the buildings, related to other existing methods. Second, it is robust against most common attacks, specially the squaring attack or line simplification.

[6] http://www.neuma.fr

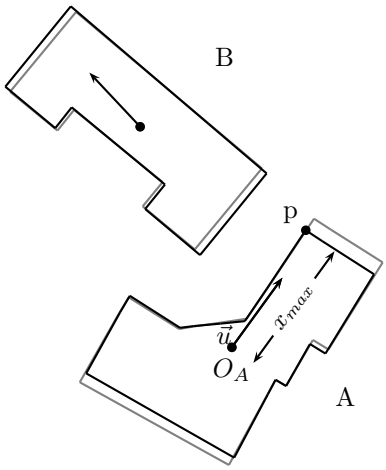


Figure 1.7: Building watermarking by oriented stretching

Organization

This document is organized as follows. The subsequent chapters dive into the details of definitions and technical results of each part: Chapter 3 presents theoretical results on query-preserving watermarking. Chapter 4 turns to its practical counterpart as developed in the Watermill software. Then, we consider watermarking methods for specific data types: XML streams (Chapter 5), symbolic music streams (Chapter 6), and geographical databases (Chapter 7). The final chapter concludes with possible developments, and Appendix A presents other studies.

2
Introduction (French)

Ce chapitre d'introduction donne une vue générale du tatouage, de ses spécificités pour les bases de données, et présente les principaux résultats obtenus.

Tatouage de documents électroniques

Informellement, le tatouage (*digital watermarking*) est une altération volontaire d'un document électronique, dans le but d'y dissimuler un message : une marque. Les applications du tatouage sont nombreuses, par exemple :

- La protection de la propriété intellectuelle (figure 2.1) : dans de nombreuses applications, les fournisseurs ou propriétaires de données ont fortement investi en temps ou en argent pour la construction des documents électroniques de grande qualité (citons par exemple l'exploration de zones géographiques pour la fabrication de données géo-localisées). Mais, en raison du caractère électronique de ces documents, des acheteurs malveillants peuvent tenter de les revendre en leur nom. Par dissimulation de l'identité du propriétaire dans les documents, le tatouage permet de prouver l'identité du propriétaire quand un document suspect est découvert. Un exemple classique est la fonctionnalité de tatouage de photographies de *Digimarc* pour le logiciel Adobe Photoshop[1].

- La traçabilité des documents, ou estampillage (*fingerprinting*) : à la place de l'identité du propriétaire, il est possible de dissimuler l'identité de l'acheteur des documents, afin de remonter à la source d'une vente illicite (figure 2.2). Citons par exemple l'identification, parmi les votants des *Academy awards*, de ceux qui ont divulgué leurs copies personnelles des films en compétition[2].

- L'incrustation de méta-données : il s'agit de dissimuler dans le document son identifiant unique, ou les paramètres techniques qui ont permis sa réalisation. Ainsi, ces méta-données restent attachées au document, quelles que soient les transformations (raisonnables) du document dans le futur.

Un protocole de tatouage requière deux algorithmes, le *marqueur* et le *détecteur*, qui dissimulent et extraient respectivement la marque. Cette marque est considérée généralement comme *invisible* : le tatouage a donc des similitudes avec le domaine de la communication cachée (*information hiding* – qui traite de la communication, sans relation avec le document qui servira de support à cette communication). La plupart des applications nécessitent également des marques *robustes* aux opérations malveillantes que pourrait réaliser un attaquant (il existe également des techniques à base de marques fragiles, mais elles ne sont pas traitées ici).

[1] `https://www.digimarc.com/solutions/`
[2] `http://www.msnbc.msn.com/id/4037016/`

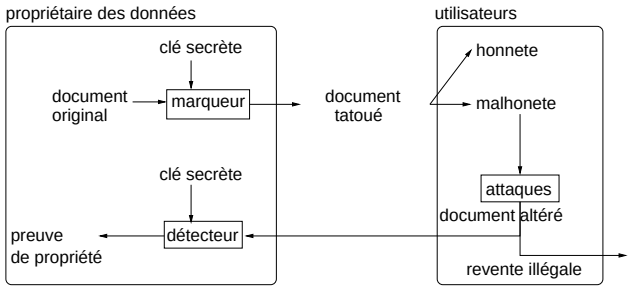


FIGURE 2.1 – Scénario classique de tatouage (*watermarking*)

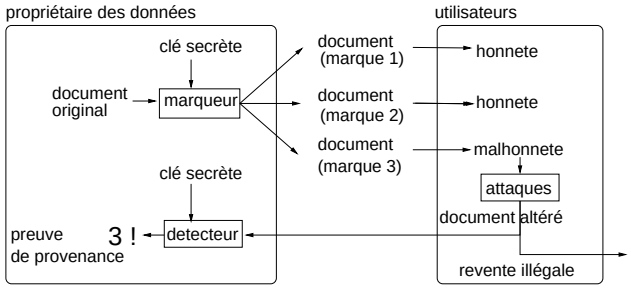


FIGURE 2.2 – Scénario classique de traçabilité, ou estampillage (*fingerprinting*)

Les contraintes usuelles des protocoles de tatouage sont les suivantes :

- Invisibilité : l'ajout d'une marque de doit pas impacter l'usage normal du document (ne doit pas diminuer sa qualité en deçà d'une limite raisonnable). Il faut cependant remarquer que le tatouage est voué à altérer le document original pour être efficace. En effet, dissimuler de l'information uniquement dans l'*encodage* du document (dans les bits inutiles par exemple) serait extrêmement sensible à un attaquant sérieux et informé. Ainsi, le tatouage doit altérer la *sémantique* du document, la difficulté résidant dans le contrôle de cette altération.

- Maîtrise de la capacité : la quantité d'information dissimulable dans un document doit être prédictible. Une capacité élevée est bien sûr souhaitable.

- Faible taux de faux-positifs : la probabilité de détecter une marque dans un document non tatoué doit être négligeable.

- Robustesse : le détecteur doit être capable de détecter la marque dans des documents raisonnablement altérés par un attaquant. La force des altérations réalisées par ce dernier est néanmoins limitée, car l'attaquant souhaite obtenir des données ayant toujours une valeur marchande, donc d'une qualité raisonnable.

- Publicité des algorithmes : la sécurité de la méthode de tatouage ne doit pas reposer sur le secret des algorithmes employés, mais seulement sur une clé privée (principe de Kerckhoffs [59, 60]).

- Sécurité : un attaquant ne doit pas pouvoir inférer le lieu du tatouage ni la clé utilisée [20].

- Méthode aveugle : le détecteur doit idéalement fonctionner sans avoir accès au document original, non tatoué.

Parmi ces nombreux critères, il est important de souligner le compromis entre la robustesse du tatouage et son invisibilité : un tatouage plus robuste nécessite une plus forte altération des données originales. L'attaquant est également limité par la contrainte d'invisibilité : son attaque (altération) doit être limitée, de façon à ce que le document reste de bonne qualité et soit ainsi revendable.

Tatouage des bases de données

Si les techniques de tatouage sont apparues initialement dans le domaine multimédia pour l'image, le son ou la vidéo, elles ont des applications naturelles en bases de données. En effet, la quantité de données structurées disponibles sur Internet est en augmentation continue, qu'il s'agisse de données publiques (données environnementales, sociétales, économiques, comme dans le projet Data Publica[3]) ou de données commerciales (statistiques de ventes, information boursière, profiles clientèle, etc.). Cette augmentation va de paire avec le besoin de protection. Il existe d'ailleurs des traces historiques d'une altération volontaire des données afin d'en protéger la propriété intellectuelle. Un exemple remarquable [57] est le choix de règles d'arrondies spécifiques par les éditeurs de tables de logarithmes sous forme papier (figure 2.3).

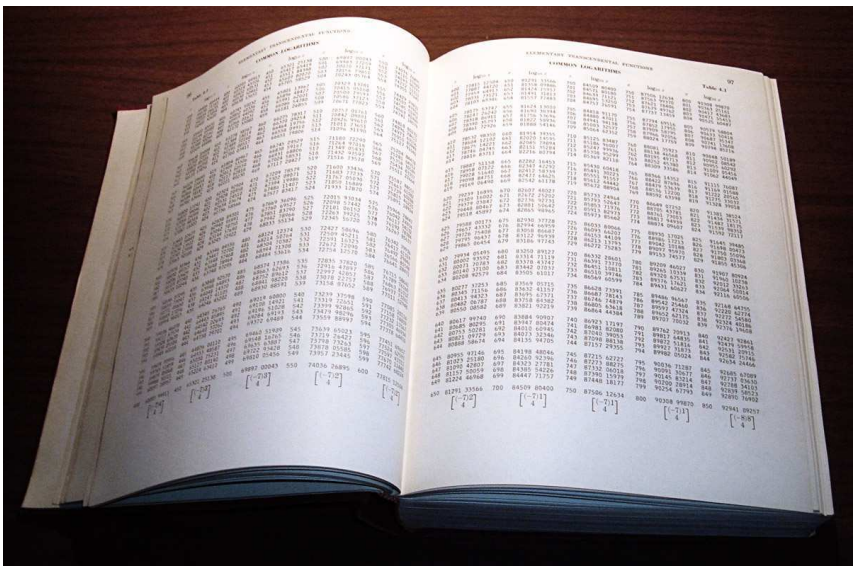


FIGURE 2.3 – Une édition de tables de logarithmes : les conventions d'arrondies furent utilisées pour la preuve de propriété (source : Wikipédia)

Les bases de données comportent cependant des spécificités importantes pour le tatouage :

- Riches interconnexions entre données : alors que les images sont des juxtapositions régulières de pixels, un jeu de données décrit des relations variées entre n-uplets, sans ordonnancement précis entre eux. Ceci pose le problème de la synchronisation du marqueur ou du détecteur avec les données, afin de localiser précisément le tatouage.

- Sémantique partagée : une base de données est généralement utilisée ou croisée avec d'autre bases de différentes provenances, comme par exemple la jointure des informations hôtelières avec une carte touristique. Il existe donc une sémantique partagée entre jeux de données, pour un nommage cohérent des informations (par exemple un même nom ou un même code pour la rue d'un hôtel et d'un site remarquable). Cette sémantique limite les possibilités de l'attaquant, car un jeu de données altéré devra respecter cette sémantique pour conserver une valeur quelconque.

[3] `http://www.data-publica.com/`

- Nombreux types de données : nombres, données catégoriques, flux, données géométriques, etc.

- Usages variés : à la différences des documents multimédia qui sont utilisés de façon directe, une base de données peut être explorées de nombreuses façons, à travers des langages de requêtes expressifs.

- Contraintes de qualité diverses : le tatouage doit respecter la qualité du document. En général, la qualité d'un document multimédia s'exprime comme une mesure globale entre le document d'origine et sa version tatouée, ou entre le document tatoué et sa version attaquée (comme par exemple le rapport signal-bruit, ou PSNR). De façon différente, les bases de données se prêtent à une formalisation des contraintes de qualité : précision des valeurs des n-uplets, impératifs de dépendances fonctionnelles entre n-uplets, et de façon générale, toute contrainte dépendant de l'application visée et exprimée idéalement dans un langage générique (contraintes d'intégrité sémantiques).

- Nombreuses modalités d'accès : les données suspectes peuvent être accessibles directement par le détecteur sous forme de relations brutes, mais également au travers de vues partielles.

- Incrémentalité : de façon générale, une base de données doit être mise à jour afin de rester précise (par exemple, une variation de 10% du jeu de données intégral peut être observée entre deux mises à jour des données géographiques nationales [91]). Ainsi, il est nécessaire de savoir maintenir ou faire évoluer le tatouage au fur et à mesure que la base de données est mise à jour chez les utilisateurs légitimes.

- Efficacité : comme les bases de données sont en général de très grande taille et enfouies dans un système de gestion standardisé, la procédure de tatouage doit y être intégrée et doit donc s'exécuter efficacement, en engendrant un faible surcoût. Cette propriété est particulièrement importante pour la vente de données à courte durée de vie, comme les données de prévisions (par exemple météorologiques ou boursières).

L'exemple suivant présente un scénario de tatouage de bases de données, où des contraintes de qualité spécifiques sont illustrées.

Exemple 1 *La relation suivante présente une instance de base de données touristique. Pour cette application, l'objectif est de dissimuler de l'information en modifiant légèrement le prix des transports proposés, tout en respectant les contraintes de qualité imposées par le propriétaire des données :*

- C_1 *: Autoriser une altération* ± 10 *sur chaque prix (price) (contrainte de précision au niveau de chaque n-uplet).*

- C_2 *: Autoriser une altération* ± 20 *sur le total global des prix (contrainte portant sur un agrégat de toute la relation, sans paramètre).*

- C_3 *: Pour* chaque *voyage (travel) proposé t, une altération* ± 10 *est autorisée sur le prix total du voyage t (contrainte d'agrégat sur une partie de la relation, dépendant d'un paramètre).*

Route :

travel	*transport*
India discovery	*T1*
India discovery	*T2*
Nepal Trek	*T1*
Nepal Trek	*T3*
Nepal Trek	*T4*
TourNepal	*T4*
TourNepal	*T5*

PriceTable :					*PriceTable′ :*	*PriceTable″ :*
transport	*departure*	*arrival*	*type*	*price*	*... price*	*... price*
T1	*Paris*	*Delhi*	*plane*	*35*	*45*	*25*
T2	*Delhi*	*Nawal.*	*bus*	*20*	*30*	*30*
T3	*Delhi*	*Kathm.*	*plane*	*15*	*25*	*5*
T4	*Kathm.*	*Simikot*	*plane*	*30*	*20*	*40*
T5	*Kathm.*	*Daman*	*jeep*	*50*	*10*	*40*
T6	*Kathm.*	*Paris*	*plane*	*10*	*10*	*10*

PriceTable représente l'instance originale du propriétaire des données. PriceTable′ et PriceTable″ sont des instances tatouées (les prix sont altérés). PriceTable′ ne respecte pas la contrainte C_3, car le coût du voyage « India discovery » est maintenant de 75 au lieu de 55. Au contraire, PriceTable″ respecte toutes les contraintes de qualité demandées.

Comme illustré dans l'exemple précédent, la recherche d'un tatouage valide respectant toute les contraintes peut s'avérer un problème combinatoire difficile, pour des contraintes générales.

Démarche de ce travail

Plusieurs propositions initiales pour le tatouage de bases de données ont été présentées indépendamment aux conférences VLDB 2002 par Rakesh Agrawal et Jerry Kiernan, SIGMOD 2003 par Radu Sion, Mickael Attalah et Sushil Prabakhar, et PODS 2003 pour le présent travail.

D'une part, Agrawal et Kiernan ont proposé une solution complète de tatouage dans le cadre de leur projet de bases de données hippocratiques [9]. Ils ne considèrent pas explicitement le problème de l'impact du tatouage sur le résultat de requêtes, mais ont observé que la moyenne et la variance des attributs numériques – prises sur l'intégralité des relations – ne sont que peu altérées par le tatouage.

D'autre part, Radu Sion et al. et le présent auteur ont proposé les premières méthodes de tatouage de bases de données intégrant la préservation du résultat de requêtes importantes[4]. La technique de Radu Sion et al. est orientée vers une solution complète et en pratique. La recherche d'un tatouage valide est réalisée par une approche d'essais et erreurs.

Le présent travail a débuté en 2001 au laboratoire Cedric du CNAM-Paris, dans l'équipe Vertigo alors dirigée par Michel Scholl (équipe actuellement dirigée par Michel Crucianu). Cette étude a commencé par les aspects théoriques du tatouage, en suivant une méthodologie courante en base de données mais avec une parfum de sécurité :

- Exprimer les contraintes de tatouage par un langage reposant sur des logiques connues.
- Décrire l'ensemble des tatouages valides, en tirant partie de la structure syntaxique des contraintes et de la structure des données. Cette description doit permettre d'obtenir des bornes inférieures sur la capacité de dissimulation d'une instance, ou des résultats d'impossibilité. En parallèle, obtenir des bornes inférieures et supérieures sur l'altération que doit subir la base de données.
- Obtenir une solution pratique avec un langage de description de contraintes simple et une optimisation du tatouage au niveau logique et physique.
- Généraliser les techniques à des types de données spécifiques.
- Et bien sûr, réaliser cela en respectant les critères usuels des protocoles de tatouage (invisibilité, grande capacité, robustesse, caractère aveugle ou non, etc.).

Cette démarche souligne les spécificités du tatouage de bases de données, comparées aux travaux classique sur le tatouage. Ces spécificités sont résumées dans la figure 2.4. Dans la suite est présentée une vue d'en-

[4]Il faut noter qu'une méthode de tatouage de graphes valués avec préservation d'une propriété unique a été proposée par Sanjeev Khanna et Francis Zane en 2000 [61]. Ce travail sera discuté en détail dans le chapitre 3.

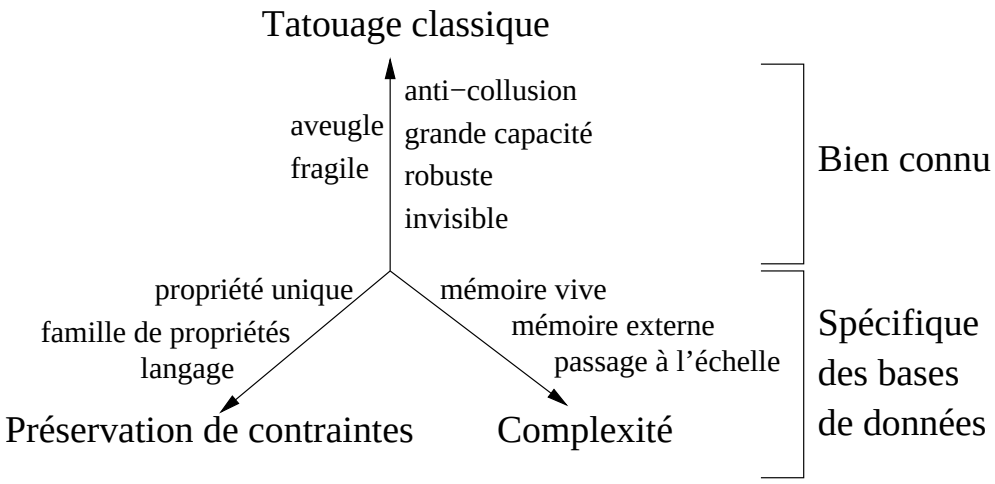


FIGURE 2.4 – Critères pour le tatouage de bases de données

semble des travaux : l'étude théorique du tatouage avec préservation de requêtes au chapitre 3, l'obtention d'un algorithme efficace au chapitre 4, et une variation autour de ces techniques pour différents domaines d'application aux chapitres 5, 6 et 7. L'ordre de présentation choisi est thématique et ne respecte pas l'exact chronologie des publications.

Chapitre 3 : tatouage avec préservation de requêtes

Le premier chapitre de ce travail concerne la modélisation de contraintes de qualité pour les bases de données numériques, au moyen de requêtes d'agrégat de somme (publication à PODS 2003 [39] et TODS 2010 [40]). Nous y considérons, en empruntant une méthodologie maintenant classique en théorie des bases de données et en complexité descriptive, la relation entre d'une part la capacité de tatouage, et d'autre part la puissance d'expression du langage de requêtes utilisé pour la spécification des contraintes. Par exemple, la contrainte C_3 de l'exemple 1 est équivalente à la préservation de la somme des prix des transports v pour tout voyage donné u, sélectionné par la requête du premier ordre

$$\psi(u,v) \equiv \exists x_1 x_2 x_3 x_4 x_5 \mathit{Route}(u,v) \wedge \mathit{PriceTable}(v, x_1, x_2, x_3, x_4, x_5).$$

Dans cette même direction, mais sans rapport avec les bases de données, Khanna et Zane [61] ont proposé un protocole de tatouage à capacité de dissimulation garantie avec préservation d'une requête *paramétrique* spécifique : la longueur du plus court chemin sur les graphes valués. Leur technique d'insertion d'information ne modifie la longueur des plus courts chemins entre n'*importe quelle* paire de sommets que dans une limite prescrite. La capacité de tatouage est en $\Omega(n^{1/2-1/d})$ bits pour une altération d des longueurs des plus courts chemins, pour tout graphe à n arêtes. Du point de vue théorique, ils observent que la requête de plus court chemin à une faible complexité, et indiquent que les protocoles de tatouage pour des propriétés NP-difficiles sont probablement difficiles à analyser.

Dans un premier temps, nous avons généralisé leur modèle aux bases de données à valeurs numériques (plus seulement les graphes valués). Nous avons également considéré la préservation de requêtes exprimées dans un langage de requêtes, et pas seulement une propriété unique comme le plus court chemin.

Pour une rapide présentation des résultats obtenus, les paramètres pertinents sont la taille de l'instance de base de données considérée, que nous notons n dans cette introduction, et la distorsion maximale sur les contraintes de qualité, notée par d. Pour simplifier et sans perte de généralité, l'altération autorisée sur chaque valeur numérique sera de +1 ou -1.

Considérons tout d'abord le problème de calculer le nombre total de tatouages possibles pour la base de données Travel de l'exemple 1, c'est-à-dire la capacité de tatouage de cette instance. Il apparaît que ce dénombrement est aussi difficile que de compter le nombre de chemins acceptants d'une machine NP :

Theoreme 2 *(informel) Calculer la capacité de tatouage est #P-complet.*

Ce constat étant fait, on s'intéresse aux bornes inférieures de cette capacité, en tenant compte du compromis avec l'altération des données.

Theoreme 3 *(informel) Pour tout ensemble fini de contraintes et toute instance de bases de données, il existe un protocole de tatouage de capacité $\Omega(d \log n)$ avec erreur au plus d sur le résultat des requêtes.*

Mais $d \log n$ bits est traditionnellement considéré comme une capacité minuscule, et l'on recherche plutôt des protocoles de tatouage à capacité n^c, pour une constante c. Un premier résultat est que, si les contraintes sont un ensemble fini de requêtes non-paramétriques (comme la contrainte C_2 de l'exemple 1), la capacité de dissimulation est importante :

Theoreme 4 *(informel) Pour tout ensemble de contraintes non-paramétriques et toute instance de base de données, il existe un protocole de capacité n avec erreur constante nulle.*

L'idée principale de la méthode est de trouver deux valeurs numériques (deux n-uplets distincts) qui impactent exactement le même sous-ensemble de requêtes, et de modifier leur valeur de façon opposée. En rappelant l'exemple 1, les n-uplets $T1$ et $T2$ impactent ensemble le voyage $India\ discovery$. Ajouter 1 au prix de $T1$ et retrancher 1 au prix de $T2$ (ou le contraire) n'a pas d'impact sur le prix complet du voyage. Chacune de ces *paires de compensation* permet de dissimuler 1 bit d'information. Si l'on considère maintenant les requêtes paramétriques (comme la contrainte C_3), qui engendrent un ensemble de contraintes complexes, trouver de telles paires de compensation est plus délicat. En effet, nous avons montré que l'impact du tatouage est illimité si aucune hypothèse n'est faite sur l'instance. En utilisant un outil courant en apprentissage (PAC learning [113]), à savoir la dimension de Vapnik-Chervonenkis [16], on peut montrer que l'ensemble des n-uplets est « pulvérisé » par les requêtes : il existe toujours une valeur du paramètre qui permet d'isoler un sous-ensemble quelconque des altération +1 ou -1. Il existe donc une valeur du paramètre rassemblant par exemple tous les +1. Leur impact cumulé permet alors de dépasser toute limite constante imposée sur l'altération du résultat d'une requête.

Theoreme 5 *(informel) Il existe une requête paramétrique et une instance de base de données qui ne possèdent pas de protocole de tatouage de capacité n^c et à erreur constante.*

Puis, nous avons utilisé des hypothèses de restriction sur les instances qui sont apparues dans d'autres contextes, comme l'étude de la complexité des données en fonction de leur degré ou de leur largeur d'arbre (qui mesure leur similarité avec un arbre). Dans ce cadre, les résultats suivants ont été obtenus :

Theoreme 6 *(informel) Il existe toujours un protocole de tatouage à capacité n^c et à erreur constante pour :*

- *les requêtes paramétriques* locales *sur les instances de degré de Gaifman borné (par exemple, les requêtes SQL sur des graphes de degré borné). Les langages locaux incluent FO (SQL), les requêtes invariantes à l'ordre ou les requêtes $\texttt{AGGR}_{\mathbb{Q}}$ (SQL avec agrégats),*
- *les requêtes paramétriques du second ordre monadique sur les arbres ou les instances à largeur d'arbre bornée (par exemple les arbres XML et des requêtes utilisant des fragments d'XPath).*

Enfin, en utilisant les résultats de Gröhe and Tùran [37], on peut démontrer que, sans ces hypothèses, aucun protocole de tatouage (de bonne capacité et à erreur constante) ne peut être obtenu. De plus, dans une publication récente [40], nous avons montré que :

Theoreme 7 *(informel)*

- *Il existe une requête paramétrique d'arité k telle que tout protocole de tatouage de capacité n^c a au moins erreur k.*
- *Pour les classes d'instances et de requêtes précitées d'arité k, il existe un protocole de tatouage de capacité n^c avec erreur au plus k. Ce protocole, que nous explicitons, est donc optimal.*

L'ensemble de ces résultats montre que, pour des bases de données naturelles, il existe une grande capacité de tatouage. Mais la solution théorique proposée ne remplie pas tous les critères usuels des protocoles de tatouage. Par exemple, l'algorithme proposé n'est pas aveugle (l'ensemble des données d'origine est requis lors de la détection). L'exploration effective des tatouages valides et l'obtention d'un algorithme (presque) aveugle est le sujet du chapitre suivant.

Chapitre 4 : aspects pratiques, le système WATERMILL

Ce chapitre concerne les aspects pratiques du tatouage, dans le but d'obtenir un système complet. Cette étude a été réalisée durant le projet ACI/ANR Tadorne[5] dont j'ai été l'initiateur et le coordinateur. Durant ce projet, la conception d'un langage de contrainte facile d'utilisation a été réalisé. Les contraintes de l'exemple 1 peuvent ainsi être exprimées :

```
(C1) local 10  on price
(C2) global 20 on (select sum from price)
(C3) forall t in (select route from travel)
          global 10 on (select sum(price) from route where travel=t)
```

L'algorithme de tatouage réalise la recherche de paires de compensation du chapitre précédent. Du point de vue de son implantation, la difficulté principale est le passage à l'échelle. La technique proposée a été de traduire l'opération de recherche de tatouage en une requête SQL unique, dont l'évaluation est relayée au SGBD. Du point de vue du tatouage, plusieurs amélioration ont été obtenues. D'abord, nous avons adapté les techniques d'Agrawal and Kiernan [7, 8] (qui ne prennent pas en compte les requêtes à préserver) pour obtenir la synchronisation avec le tatouage et sa sécurité. Nous avons obtenu un meilleur caractère aveugle de l'algorithme en remplaçant la méthode simple des paires de compensation (qui nécessite les données originales pour la détection), par la recherche de bits de valeur opposée dans les données elles-mêmes (exemple 8).

Exemple 8

Compensation de la valeur absolue (non-aveugle)

transport	*price (original)*	*price (tatoué)*
T1	*35*	*25 (-10)*
T2	*45*	*55 (+10)*
total	*80*	*80*

Compensation de bits opposés (aveugle aux données)

transport	*price (original)*	*price (tatoué)*
T1	$(35)_2 = 10\,\boxed{0}\,011$	$10\,\boxed{1}\,011 = 43$
T2	$(45)_2 = 10\,\boxed{1}\,101$	$10\,\boxed{0}\,101 = 37$
total	*80*	*80*

La méthode est semi-aveugle, ou aveugle aux données : l'ensemble des données originales n'est plus nécessaire, mais l'ensemble des positions de tatouage doit être mémorisé et produit lors de la détection (comme pour la méthode de Sion et al. [103]). L'algorithme met également en oeuvre un code anti-collusion sophistiqué du à Tardòs [109]. La méthode complète [26,67] obtenue avec Julien Lafaye, Camélia Constantin et Meryem Guerrouani, a été publié dans TKDE 2008. Elle a été implantée dans le logiciel GPL WATERMILL[6], et a été validée sur de grands jeux de données. Il a été possible par exemple de tatouer 1 000 000 de n-uplets tout en préservant 100 contraintes en quelques minutes, alors que les méthodes concurrentes nécessitent plusieurs heures ou jours.

[5] `http://ufrsciencestech.u-bourgogne.fr/~gadavid/tadorne/`
[6] `http://watermill.sourceforge.net`

Chapitres 5, 6 et 7 : algorithmes spécifiques

La suite de ce travail a été la conception d'algorithmes spécifiquement adaptés à certains types de données, au delà des données numériques. Dans ce cadre, le squelette des algorithmes utilisés est toujours la méthode d'Agrawal et Kiernan, et la contribution réside dans l'adaptation à des applications précises : les flux XML, les bases de données musicales symboliques, les bases de données géographiques et les bases de données multimédia. La suite de ce document présente les trois premiers types.

Chapitre 5 : flux Xml typés

La première méthode spécifique concerne le tatouage de flux XML. Dans le cadre classique d'étude des flux à haut débit, les données doivent être traitées en mémoire constante : toute opération comme la validation de type ou la transformation de données doit être réalisée par un automate fini (figure 2.5). Dans ce travail mené avec Julien Lafaye et présenté à DbSec 2006 [66], nous avons considéré le problème du tatouage de flux XML tout en préservant leur type, exprimé par une DTD non-récursive. Le validation de tels types peut être réalisée par un automate fini [101]. Nous tatouons ces flux par une recopie du flux durant sa vérification de type, mais en empruntant parfois un *détour* dans une exécution équivalente de l'automate. L'erreur réalisée est contrôlée par la distance d'édition entre le flux d'origine et sa version tatouée.

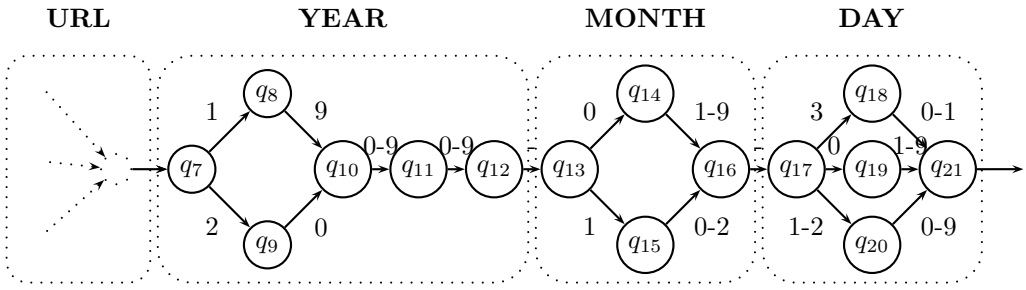


Figure 2.5 – Spécification partielle d'un type de flux pour un élément `date`

Chapitre 6 : partitions musicales symboliques

Un second résultat a été obtenu lors de mon arrivée au laboratoire Le2i, dans l'équipe SISI dirigée par Kokou Yetongnon. Après avoir développé une méthode de tatouage pour les données multimédia avec Richard Chbeir [21], je me suis tourné vers les bases de données musicales. Ce travail a été réalisé dans le cadre du projet Neuma[7], également financé par l'ANR, initié et coordonné par Philippe Rigaux. La musique y est considérée dans une représentation symbolique : non comme un fichier son ou comme l'image d'une partition musicale, mais comme la description précise des notes, rythmes et annotations. Un exemple en est le format MusicXML. Nous avons proposé dans le système Neuma un outils de tatouage de telles partitions musicales symboliques. Dans notre approche, une partition est vue comme un flux de notes. Comme cible du tatouage, nous avons sélectionné une annotation utile pour les musiciens débutants, les annotations de doigté (figure 2.6). En clair, une annotation de doigté précise quel doigt de la main utiliser pour la réalisation de chaque note, et plusieurs doigtés sont possibles pour une même partition. D'un point de vue informatique, la qualité d'un doigté dépend de la difficulté à jouer ce doigté, qui est reliée aux capacités physiques de la main (plusieurs travaux modélisent cette capacité). L'idée principale est de tatouer la partition annotée en favorisant un doigté particulier parmi tous les doigtés possibles. Les doigtés originaux sont en général réalisés par des experts et sont de grande qualité. L'objectif est donc de produire des doigtés tatoués avec une altération contrôlée par rapport aux doigtés d'origine. Cette méthode, obtenue avec Philippe Rigaux, Lylia Abrouk et Nadine Cullot, a été présentée à ISMIR 2009 [43].

[7] http ://www.neuma.fr

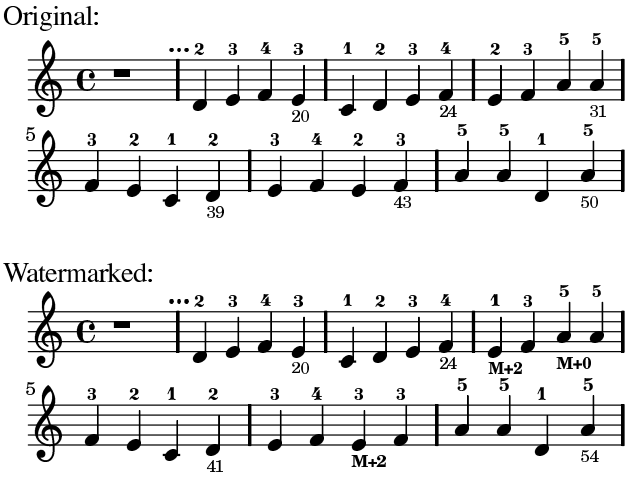


FIGURE 2.6 – Une partition originale avec une annotation de doigté de grande qualité, et sa version tatouée. Les annotations de doigté figurent au dessus de la portée (main droite, 1 : pouce, ..., 5 : auriculaire). En dessous figure le coût physique pour jouer ce doigté. La version tatouée est plus difficile à jouer, avec trois positions altérées indiquées par un M.

Chapitre 7 : données géographiques

Ce dernier résultat concerne les bases de données géographiques, et a été obtenu également durant le projet Tadorne en partenariat avec le laboratoire Cogit (IGN – Institut géographique national), avec Julien Lafaye, Jean Béguec et Anne Ruas. Il a été présenté à SSTD 2007 [65]. Nous avons proposé une solution de tatouage par les cartes vectorielles utilisées pour leur précision, avec un accent sur la couche dite du bâti (les bâtiments – la plus grande part des bases de données géographiques professionnelles). La difficulté nouvelle par rapport à l'existant est de trouver un identifiant raisonnable pour chaque polygone, et de prendre en compte les métriques de qualité spécifiques utilisées dans les applications géographiques. Dans ce travail, nous avons supposé que les données sont référencées suivant une norme commune à tous les participants (le WGS 84 du système GPS par exemple). Puis nous avons construit des identifiants robustes de polygones en prenant les bits de poids fort de leur centroïde (un attaquant doit modifier énormément la position d'un bâtiment pour modifier cet identifiant). Enfin, la dissimulation de données a été réalisée par étirement du bâtiment selon sa direction principale, comme indiqué dans la figure 2.7.

Cette méthode très simple possède de bonnes propriétés, et a un faible impact sur la qualité angulaire des bâtiments, par rapport aux autres méthodes existantes. De plus, elle est robuste aux attaques courantes, en particulier l'attaque par équarissage (visant à rendre droits les angles presque droits) ou la simplification de lignes (cherchant à supprimer des points parmi ceux qui sont presque alignés).

Plan

Ce document est organisé de la façon suivante. Les chapitres suivants plongent dans les détails des définitions et des résultats techniques de chaque partie : le chapitre 3 présente les résultats théoriques sur le tatouage avec préservation de requêtes. Le chapitre 4 présente une version utilisable en pratique telle qu'elle figure dans le logiciel WATERMILL. Enfin, nous considérons le tatouage pour des types de données spécifiques : les flux XML (chapitre 5), les bases de données musicales symboliques (chapitre 6) et les bases de données géographiques (chapitre 7). Le dernier chapitre conclue avec des pistes de développement et l'appendice A résume nos autres travaux, non présentés ici.

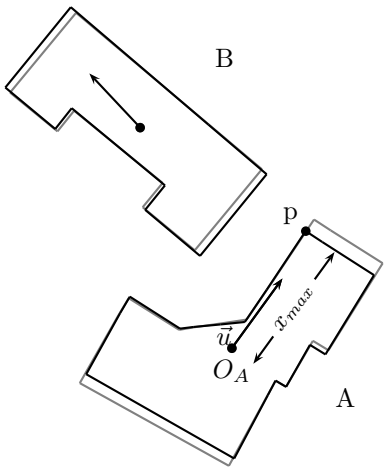


FIGURE 2.7 – Tatouage d'un bâtiment par étirement

3

Numerical database watermarking with query preservation

In this chapter we consider database watermarking where the result of valuable queries has to remain invariant. We define the notion of query-preserving watermarking protocols. We then distinguish between two approaches: the view model, where the data owner has to prove ownership using only results of queries, and the instance model, where a full access to the suspect instance is provided. We also consider adversarial and non-adversarial scenarios.

Query-preserving watermarking We focus on the watermarking of databases, in the general setting where data servers perform queries in a language $\mathcal{L}$. The data owner has a valuable database instance, and data servers apply for a copy of this database, providing queries $\psi_1, \ldots, \psi_k$ they will answer to final users. These queries will be parametrized by final users'inputs. The problem is then to construct a *query-preserving watermarking protocol* that respects the following conditions:

- the protocol hides a message m into the owner's database instance, and induces a guaranteed distortion on the results of queries $\psi_1(\bar{a}), \ldots, \psi_k(\bar{a})$, for any user input $\bar{a}$.
- the protocol is able to extract the message hidden in a suspect data set, based on answers to queries $\psi_1, \ldots, \psi_k$ only (the owner acts as any final user to get these answers).
- the protocol fails extracting messages from unwatermarked databases.
- the protocol is robust against reasonable attacks, like random distortion of query answers.
- the protocol has a scalable capacity: the amount of hidden bits grows with the database size.

Such a watermarking protocol is driven by what is important to the final user: results of queries. Notice that data servers may answer other queries to users, but only distortion on $\psi_1, \ldots, \psi_k$ is guaranteed.

Example 9 *Tables below present a touristic database instance. The goal is to hide information by slightly modifying transport prices so that the total cost of any travel remains unchanged. This is equivalent to preserving the following aggregate query, for any value of parameter t:*

$$\psi(t) \equiv \texttt{select sum(price) from PriceTable, Route}$$
$$\texttt{where PriceTable.transport} = \texttt{Route.transport}$$
$$\texttt{and travel} = t.$$

Route:

travel	*transport*
India discovery	*T1*
India discovery	*T2*
Nepal Trek	*T1*
Nepal Trek	*T3*
Nepal Trek	*T4*
TourNepal	*T4*
TourNepal	*T5*

PriceTable:

transport	*departure*	*arrival*	*type*	*price*
T1	*Paris*	*Delhi*	*plane*	*35*
T2	*Delhi*	*Nawalgarh*	*bus*	*20*
T3	*Delhi*	*Kathmandu*	*plane*	*15*
T4	*Kathmandu*	*Simikot*	*plane*	*30*
T5	*Kathmandu*	*Daman*	*jeep*	*50*
T6	*Kathmandu*	*Paris*	*plane*	*10*

Watermarking capacity The fundamental difficulty in query-preserving watermarking is to obtain a protocol with a reasonable bandwidth. For example, if n bits can be hidden in a data set, 2^n distinct messages can be hidden. By mapping distinct messages with distinct data servers, the message extraction allows to identify the suspect server that performed a copy. Another application is to combine the hidden message with an error-correcting code in order to increase its robustness against attacks. Hence, obtaining lower bounds on the watermarking capacity is interesting.

In this direction, [61] proposed a protocol with guaranteed capacity for a specific *parametric* query: shortest path queries on weighted graphs. Their information insertion does not modify the length of shortest path between *any* pair of vertices beyond an acceptable distortion. The watermarking capacity is $\Omega(n^{1/2-1/d})$ for distortion d, for any graph with n edges. From the theoretical point of view, they observe that shortest path queries have a low computational complexity, and suspect that watermarking protocols for NP-hard search spaces are difficult to analyze.

Contribution: watermarking and learning theory In this chapter, we study the capacity of watermarking protocols that preserve any set of predefined queries from a language $\mathcal{L}$. Up to our knowledge, this is the first work dealing with this problem in the database theory perspective. Our goal is *not* to propose a new watermarking algorithm for databases, but to study the potential watermarking bandwidth that can be awaited from given data sets and queries (the present work has nevertheless a practical counterpart [67] as explained in the related work).

Our first contribution is an extension of [61]'s graph watermarking model into the relational setting. This natural extension requires some care, as a general query on a general structure is much more intricate than properties considered in their work (shortest paths on graphs, which are simply subsets of the graph). We then study the complexity of computing the exact watermarking capacity, which turns out to be intractable.

Our first main result shows that the difficulty of query-preserving watermarking is linked to the *informational complexity* of sets defined by queries, rather than their computational complexity. This is related to an important combinatorial parameter in computational learning theory, the *Vapnik-Chervonenkis* dimension of sets (or VC-dimension, see for example [16]). Roughly speaking, a set is said to be shattered if all its subsets can be defined by a query, and the VC-dimension represents the largest cardinality of such shattered sets. It is well known that a finite VC-dimension for a family of sets is equivalent to its learnability (in the PAC learning model [113]). In summary, our result states that if the VC-dimension is not bounded but is maximal on all input structures, that is, if any set is shattered by queries, no watermarking protocol can be obtained.

[37] showed that the VC-dimension of sets defined by first-order logic and monadic second-order logic is bounded on restricted classes of structures, and this characterization is, in some sense, optimal. These restrictions concern bounding the degree of the Gaifman graph of the structure, or bounding its tree-width, which measures its similarity with trees. This latter restriction has also fruitful applications in both database theory and computational complexity (see for example [31]).

Our second main result shows that under the same restrictions, a watermarking protocol with guaranteed capacity can be obtained. First, we construct a watermarking protocol for database instances with bounded degree Gaifman graph, preserving any *local* query. Local languages contain particularly first-order logic, order-invariant queries [36], and relational $\texttt{AGGR}_{\mathbb{Q}}$ queries [49], that expresses mostly plain `SQL` by adding grouping and aggregate functions to relational calculus [44,74]. Second, we provide a watermarking protocol

for first-order and monadic second-order queries on trees or tree-like structures. Monadic second-order logic (MSO) is of a special interest, since it is commonly used to model *pattern queries* on labeled trees, as a formal query language for XML documents (see for example [81]).

Finally, on structures with unbounded degree Gaifman graph, one can construct a first-order formula that defines sets with unbounded and maximal VC dimension. There also exists an MSO-formula yielding such sets on structures with unbounded tree-width. For both, no query-preserving watermarking protocol can be obtained. This maps a rather complete panorama of query-preserving watermarking.

Organization The chapter is organized as follows: we first give basic definitions on query-preserving watermarking on Section 3.1. We show on Section 3.2 that computing the exact watermarking capacity is hard, and that no watermarking protocol can be obtained for unrestricted database instances, even for trivial queries. We then exhibit restrictions on database instances and query languages that allow watermarking with a reasonable amount of hidden information: local languages on structures with bounded degree Gaifman graph on Section 3.3, monadic second-order queries on trees and tree-like structures on Section 3.4. Section 3.5 extends the previous results so that malevolent attacks are taken into account. Complementary results on easier database watermarking scenarios are given in Section 3.6, and the incremental updatability of watermarked instances is exposed in Section 3.7. Finally, Section 3.8 compares the present method with its practical counterpart, and the last section concludes.

Related work A wide part of the watermarking literature focuses on multimedia data including images, sound and video [28, 57, 120]. The watermarking capacity is a classical topic in multimedia watermarking, where the common tool is information theory [19]. But this community does not consider intricate query answers on structured data sets, but rather global metrics like peak signal-to-noise ratio (PSNR) on a signal (an image) without formal structure.

Beside works cited in the introduction [7, 61], watermarking of structured data like trees, graphs, or solutions of an optimization problem are studied in [88, 89, 116, 118]. These approaches do not consider the notion of queries. [104] and [39] introduced query-preservation methods for watermarking. [104] handle potentially any kind of constraints by a greedy procedure calling external checking programs (*usability plugins*). [102] extend this work so that better watermarks are obtained. These very general methods do not elaborate on the syntactical form of queries in order to find valid watermarks. [39] and [67] take into account the syntactic form of aggregate constraints, so that the watermark search procedure is optimized.

Our formalism follows the approach of [61]. In this work, a unique parametric query is considered, namely shortest paths in weighted graphs. This formal approach was first generalized in [39]. In this latter work, a lower bound on the watermarking capacity is obtained for queries taken from several query languages, along with an upper bound on data distortion. No lower bound on data distortion was provided. In the present work, we propose such a lower bound. We also provide a watermarking protocol achieving this minimal error. Hence this bound is tight. Finally, even without a malevolent attacker, the previous method was probabilistic, while the method presented in this chapter is deterministic. We also provide results on different scenarios, where the data owner has full access to the suspect instance, and when non-parametric queries are to be preserved.

3.1 Query-preserving watermarking

3.1.1 Basic definitions

Weighted structures In this work, attention is focused on watermarking numerical data for the sake of simplicity. Generalization to other domains with a distance function (for example strings with a similarity measure, or a semantic distance) can be considered. The data owner distinguishes those numerical values that can be modified for watermarking, denoted in the sequel by *weights*. Our watermarking protocols will modify weights of a database instance, while leaving other values unchanged. We model such database instances by *weighted structures* [35]. Let τ be a signature (or database schema), that is a finite set of

relation symbols $\{\mathbf{R_1}, \ldots, \mathbf{R_t}\}$, with respective arity $r_1, \ldots, r_t$. A finite structure $\mathcal{G} = \langle \mathcal{U}, R_1, \ldots, R_t \rangle$ (or database instance) is an interpretation of each relation symbol of the schema τ on a finite universe $\mathcal{U}$. We denote by $STRUCT[\tau]$ the set of all τ-structures. For a given $s \in \mathbb{N}$, a *weighted structure* $(\mathcal{G}, \mathcal{W})$ is defined by a finite structure $\mathcal{G}$ and a *weight function* $\mathcal{W}$, which is a partial function from $\mathcal{U}^s$ to $\mathbb{N}$, that maps a s-tuple $\bar{b}$ to its weight $\mathcal{W}(\bar{b})$.

Example 10 *The owner of the previous instance (Example 9) chooses a unique weight attribute "price". The corresponding structure is* $\mathcal{G} = \langle \mathcal{U}, Route, PriceTable \rangle$*, with, as an example of possible tuples* $(TourNepal, T4) \in Route, (T1, Paris, Delhi, plane) \in PriceTable$ *and* $\mathcal{W}(T1) = 35$.

Measure of distortion Our watermarking algorithms will introduce perturbations into the structure's weight function $\mathcal{W}$, and these perturbations must be under control of the data owner, so that the data set quality is preserved. The finite part of the weighted structure will remain unchanged: hence watermarking is equivalent to mapping the weighted structure $(\mathcal{G}, \mathcal{W})$ to a new structure $(\mathcal{G}, \mathcal{W}')$.

First, we limit the alteration of weights. Given a constant $c \in \mathbb{N}$, a weighted structure $(\mathcal{G}, \mathcal{W}')$ is said to be a *c-local distortion* of $(\mathcal{G}, \mathcal{W})$ if and only if for all $\bar{b} \in \mathcal{U}^s$, $|\mathcal{W}'(\bar{b}) - \mathcal{W}(\bar{b})| \leq c$.

It is sometimes useful to view the watermarked structure has the addition of the original structure plus a watermark, or a mark for short. We denote a mark by a function $\delta : \mathcal{U}^s \to \mathbb{Z}$. A watermarked structure $(\mathcal{G}, \mathcal{W}')$ of $(\mathcal{G}, \mathcal{W})$ is said to contain mark δ if, $\forall \bar{b} \in \mathcal{U}^s, \mathcal{W}'(\bar{b}) = \mathcal{W}(\bar{b}) + \delta(\bar{b})$.

Second, we take into account the impact of distortions on the result of queries. As a query language we will consider for example first-order (FO) or monadic second-order (MSO) formulas. First-order formulas are built from atomic formulas on the database schema with equality, and are closed under classical boolean connectives $\wedge, \vee, \neg$ and quantifiers $\exists, \forall$. In monadic second-order logic, quantification is also on sets of elements.

Since queries are performed by data servers upon final users'inputs, we are interested in parametric queries. A formula with parameter $\bar{u}$ is a formula $\psi(\bar{u}, \bar{v})$ with two distinguished variable vectors, $\bar{u}$ and $\bar{v}$, with arity r and s respectively. Given a structure $\mathcal{G}$, let $\psi(\bar{a}, \mathcal{G}) = \{\bar{b} \in \mathcal{U}^s | \mathcal{G} \models \psi(\bar{a}, \bar{b})\}$. Variables $\bar{u}$ can be assigned to a value $\bar{a}$ by a final user wishing to obtain the result of $\psi(\bar{a}, \mathcal{G})$.

Let $\psi(\bar{u}, \bar{v})$ be the parametric query to be preserved (we focus on the preservation of a unique query ψ, but extension to several queries $\psi_1, \ldots, \psi_k$ is straightforward by projection techniques). The weight of a tuple extends to the weight of a query: given a weighted structure $(\mathcal{G}, \mathcal{W})$ and a parametric query $\psi(\bar{u}, \bar{v})$, its *weight* $\mathcal{W}(\psi(\bar{a}, \mathcal{G}))$ for parameter $\bar{a}$ is the sum of weights of its tuples:

$$\mathcal{W}(\psi(\bar{a}, \mathcal{G})) = \sum_{\bar{b} \in \psi(\bar{a}, \mathcal{G})} \mathcal{W}(\bar{b}).$$

Example 11 *For the database instance in Example 9, we consider the first-order query*

$$\psi(u, v) \equiv \exists x_1 x_2 x_3 Route(u, v) \wedge PriceTable(v, x_1, x_2, x_3),$$

extracting, given a travel u, the set of transports v part of this travel. Summing their corresponding weights models the total cost of travel u. We obtain the following weights:

$$\mathcal{W}(\psi(India\ discovery, \mathcal{G})) = \mathcal{W}(T1) + \mathcal{W}(T2) = 35 + 20 = 55,$$
$$\mathcal{W}(\psi(Nepal\ Trek, \mathcal{G})) = 35 + 15 + 30 = 80,$$
$$\mathcal{W}(\psi(TourNepal, \mathcal{G})) = 30 + 50 = 80.$$

These weights corresponds to answers of the parametric aggregate query of Example 9.

Query preservation can now be defined. Let $d \in \mathbb{N}$. A weighted structure $(\mathcal{G}, \mathcal{W}')$ is a *d-global distortion* of a structure $(\mathcal{G}, \mathcal{W})$ with respect to ψ if and only if, for all $\bar{a} \in \mathcal{U}^r$,

$$|\mathcal{W}'(\psi(\bar{a}, \mathcal{G})) - \mathcal{W}(\psi(\bar{a}, \mathcal{G}))| \leq d.$$

Example 12 *We consider the original instance given in Example 9 and the same query ψ. Let $PriceTable'$ and $PriceTable''$ be two possible distortions of $PriceTable$:*

Route:

travel	*transport*
India discovery	*T1*
India discovery	*T2*
Nepal Trek	*T1*
Nepal Trek	*T3*
Nepal Trek	*T4*
TourNepal	*T4*
TourNepal	*T5*

PriceTable:

transport	*departure*	*arrival*	*type*	*price*
T1	*Paris*	*Delhi*	*plane*	*35*
T2	*Delhi*	*Nawal.*	*bus*	*20*
T3	*Delhi*	*Kathm.*	*plane*	*15*
T4	*Kathm.*	*Simikot*	*plane*	*30*
T5	*Kathm.*	*Daman*	*jeep*	*50*
T6	*Kathm.*	*Paris*	*plane*	*10*

$PriceTable'$:

...	*price*
	45
	30
	25
	20
	10
	10

$PriceTable''$:

...	*price*
	25
	30
	5
	40
	40
	10

$PriceTable'$ is a c-local distortion of $PriceTable$ for constant $c = 10$, but not a d-global distortion with respect to ψ for $d = 10$, because

$$\mathcal{W}(India\ discovery, PriceTable') = 75,$$

and

$$\mathcal{W}(India\ discovery, PriceTable) = 55.$$

$PriceTable''$ respects both local and global distortions for $c = 10$ and $d = 10$.

3.1.2 Watermarking protocols and the non-adversarial view model

The goal of query-preserving watermarking is, given an original structure from a class of structures and a query ψ, to produce altered structures that respect the prescribed local and global distortions.

Definition 13 *A watermarking problem is a tuple $(\mathcal{K}, \psi)$, where $\mathcal{K}$ is a class of weighted structures and ψ is a parametric query.*

We generalize definitions of [61] for watermarking structured data. A watermarking protocol is a pair of algorithms $\mathcal{M}$ and $\mathcal{D}$, standing for the "marker" and the "detector", respectively. Marker $\mathcal{M}$ takes as input an original structure and a binary message m to be hidden in the data, and computes the watermarked version of the original structure. This structure must preserve the result of a prescribed query ψ. Detector $\mathcal{D}$, given a suspect structure $\mathcal{G}^*$ as input, extracts the hidden message m. In the first part of this chapter, we focus on the non-adversarial model, where the suspect server does not alter its data set in order to evade detection (the full, adversarial model will be considered in Section 3.5). However, even in the non-adversarial model, constructing correct structures preserving ψ is a combinatorial problem on its own.

Definition 14 *Given a watermarking problem $(\mathcal{K}, \psi)$, and $l, d \in \mathbb{N}$, a (l, d)-watermarking protocol preserving ψ in the non-adversarial model is a pair of algorithms $\mathcal{M}$ and $\mathcal{D}$ such that:*

1. *$\mathcal{M}$ takes as input an original structure $(\mathcal{G}, \mathcal{W}) \in \mathcal{K}$ and a boolean message $m \in \{0,1\}^l$ and outputs a 1-local distortion $\mathcal{G}_m = (\mathcal{G}, \mathcal{W}_m) \in \mathcal{K}$ such that $\mathcal{G}_m$ is a d-global distortion of $(\mathcal{G}, \mathcal{W})$.*
2. *Algorithm $\mathcal{D}$, given as input the original structure $(\mathcal{G}, \mathcal{W})$ and the unchanged suspect structure $\mathcal{G}^* = \mathcal{G}_m$, outputs the hidden message m.*

Parameter l stands for the number of bits to be hidden. Value d is the maximum acceptable global distortion on structures produced by the marker. Note that the suspect structure $\mathcal{G}^*$ is exactly $\mathcal{G}_m$, as we consider first the non-adversarial model. Note also that we restrict our attention to 1-local distortions: weights will only be incremented (+1) or decremented (-1). In fact, we will focus on the asymptotic behavior of the number of distinct watermarked structures. There is at most $(2c+1)^{|\mathcal{U}^s|}$ different c-local distortions of $(\mathcal{G}, \mathcal{W})$, and restricting to $c = 1$ does not alter this number drastically.

The definition is still not satisfactory. Indeed, the above-defined detector has full access to the suspect structure $\mathcal{G}^*$ during detection. We call this easier case the *instance model* (considered in Section 3.6). In a more realistic scenario, the data owner can only access suspect data by acting as a simple data user. By providing parameters $\bar{a}$ to a suspect server, the owner will obtain answers to the query $\psi(\bar{a}, \mathcal{G}^*)$. We call this model the *view model*. In the sequel we denote by $\mathcal{A}(\psi(\bar{a}, \mathcal{G}))$ the set of answers to query ψ on parameter $\bar{a}$, along with their weights:

$$\mathcal{A}(\psi(\bar{a}, \mathcal{G})) = \{(\bar{b}, \mathcal{W}(\bar{b})) | \bar{b} \in \psi(\bar{a}, \mathcal{G})\}.$$

Example 15 *Recalling Example 12, we have*

$$\mathcal{A}(\psi(\textit{India discovery}, \textit{PriceTable'})) = \{(T1, 45), (T2, 30)\}.$$

The set of all possibles answers is defined by $\mathcal{A}(\psi, \mathcal{G}) = \{(\bar{a}, \mathcal{A}(\psi(\bar{a}, \mathcal{G}))) | \bar{a} \in \mathcal{U}^r\}$. This yields the complete definition:

Definition 16 *Given a formula ψ, and $l, d \in \mathbb{N}$, a (l, d)-watermarking protocol preserving ψ* in the non-adversarial view model *is a (l, d)-watermarking protocol preserving ψ in the non-adversarial model such that the detector uses only $\mathcal{A}(\psi, \mathcal{G}^*)$ as input.*

Definition 17 *A watermarking problem $(\mathcal{K}, \psi)$ is said to have a (l, d)-watermarking protocol if there exists $l, d \in \mathbb{N}$ and a pair $(\mathcal{M}, \mathcal{D})$ that is a (l, d)-watermarking protocols preserving ψ for structures in $\mathcal{K}$.*

3.1.3 Scalable watermarking protocols

Capacity and active tuples Of course, one is interested in protocols with a large hiding capacity. The largest message size l is clearly bounded by the size of the structure. But since the suspect data set is only available through queries to a server, watermarks should be hidden into weights of active tuples of the data set, that is in those tuples that are part of a query answer. Otherwise the detector has no way to recover their weights and to extract the hidden message.

Let $W^{\mathcal{G},\psi}$ be the *active tuples of* $(\mathcal{G}, \mathcal{W})$ with respect to ψ:

$$W^{\mathcal{G},\psi} = \bigcup_{\bar{a} \in \mathcal{U}^r} \psi(\bar{a}, \mathcal{G}).$$

We will often use notation W for short.

Example 18 *Recalling Example 9, active tuples are $W = \{T1, T2, T3, T4, T5\}$, and $T6$ is inactive.*

In the rest of this chapter, *we will only distort weights of active tuples.* As a consequence, there will be at most $|W|$ useful weights to modify, and the maximum watermarking capacity is $|W|$. *It is noteworthy that $\psi(\bar{a}, \mathcal{G})$ and W do not depend on the weight function $\mathcal{W}$: we can perturb $\mathcal{W}$ without modifying $\psi(\bar{a}, \mathcal{G})$ or W.* Indeed, function $\mathcal{W}$ is not part of the vocabulary of ψ.

Scalable watermarking protocols A watermarking problem may have a watermarking protocol for a constant value of l. The interesting situation is when l is an increasing function of $|W|$, i.e. the number of hidden bits grows with the number of active tuples of the problem. The best situation would be to hide $|W|$ bits of data, without distorting results of queries at all.

Definition 19 *A watermarking problem $(\mathcal{K}, \psi)$ possesses a* scalable watermarking protocol *if there exists $0 < q \leq 1$ and a constant $d \in \mathbb{N}$ such that the same pair of algorithms $(\mathcal{M}, \mathcal{D})$ is a $(\Omega(|W|^q), d)$-watermarking protocol for $(\mathcal{K}, \psi)$.*

3.2 General case

3.2.1 Computing the watermarking capacity

Let $(\mathcal{K}, \psi)$ be a watermarking problem, and $d \in \mathbb{N}$. Given a weighted structure in $\mathcal{K}$, we consider the problem of counting the exact number of d-global distortions with respect to ψ. We call this problem $\#\textsc{Mark}(\leq d)$. We do not know the exact time complexity of $\#\textsc{Mark}(\leq d)$ on the class of all structures, but we show that a simple variation of this problem is as hard as computing the number of accepting paths of any NP Turing machine, hence is complete for the classical complexity class $\#P$ [112]. In the following, we restrict our attention to 1-local distortions that entail a $+1$ alteration on an initial segment of the active tuples of size k. For a given watermarking problem $(\mathcal{K}, \psi)$ and $d \in \mathbb{N}$, we call $\#k\textsc{Mark}(\leq d)$ the number of such structures with global distortion at most d:

Definition 20 ($\#k\textsc{Mark}(\leq d)$ PROBLEM)

- *INPUT: a weighted structure $(\mathcal{G}, \mathcal{W})$ in $\mathcal{K}$.*
- *OUTPUT: the number of 1-local, d-global distortions $(\mathcal{G}, \mathcal{W}')$ of $(\mathcal{G}, \mathcal{W})$ such that, for $W = \{\bar{b}_1, \ldots, \bar{b}_k, \ldots\}$:*

$$\forall i \in \{1, \ldots, k\}, \mathcal{W}'(\bar{b}_i) = \mathcal{W}(\bar{b}_i) + 1.$$

We prove the following result:

Theorem 21 *For every $d \in \mathbb{N}$, $\#(d+1)\textsc{Mark}(\leq d)$ is $\#P$-complete on the class of all structures.*

We start by proving the NP-completeness of a class of related problems, that we call $\textsc{Bipartite-Mark}(d)$, for $d \in \mathbb{N}$. Let $G = (V_1, V_2, E)$ be a bipartite graph. A mark δ for G is a map from V_2 to $\{-1, 0, 1\}$. For a given mark δ, the distortion $\Delta(v)$ of a vertex $v \in V_1$ is the sum of alterations of its neighbors, that is:

$$\Delta(v) = \sum_{v' \in V_2 : E(v,v')} \delta(v').$$

Definition 22 ($\textsc{Bipartite-Mark}(d)$ PROBLEM)

- *INPUT: a bipartite graph $G = (V_1, V_2, E)$ where $V_2 = \{v_{21}, v_{22}, \ldots, v_{2(d+1),\ldots}\}$.*
- *OUTPUT: true if there exists a mark δ such that:*
  - *for all $i \in \{1, \ldots, d+1\}$, $\delta(v_{2i}) = +1$;*
  - *for all $v \in V_1$, $|\Delta(v)| \leq d$.*

Lemma 23 *For every $d \in \mathbb{N}$, $\textsc{Bipartite-Mark}(d)$ is NP-complete.*

PROOF. For a given $d \geq 1$, we proceed by reduction from 3SAT. Let $x_1, \ldots, x_n$ be a set of boolean variables and $\{C_1, \ldots, C_N\}$ be a set of clauses with exactly three literals. We will encode this 3SAT instance into a bipartite graph $G = (V_1, V_2, E)$. The set V_1 is composed of:

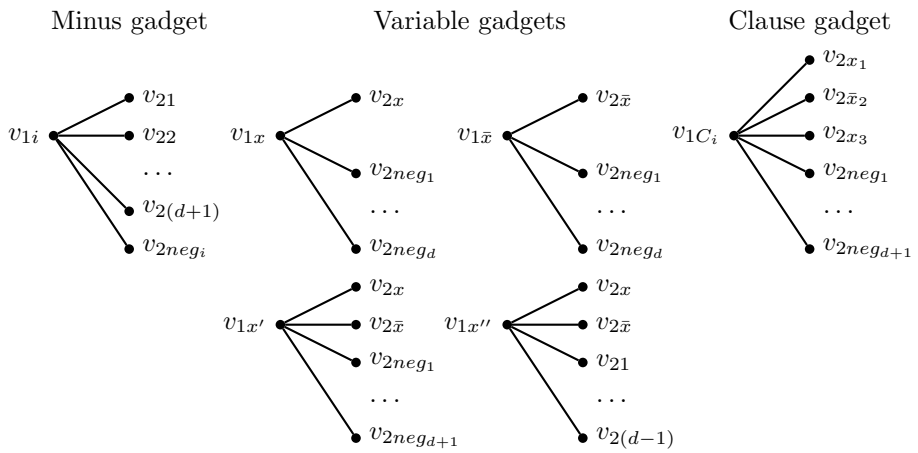


Figure 3.1: Gadgets for 3SAT reduction

- vertices $v_{11}, v_{12}, \ldots, v_{1(d+1)}$,
- vertices $v_{1x_i}, v_{1x'_i}, v_{1x''_i}, v_{1\bar{x}_i}$ for each variable x_i in the 3SAT instance,
- vertex v_{1C_i} for each clause C_i in the 3SAT instance.

The set V_2 is composed of:

- vertices $v_{21}, v_{22}, \ldots, v_{2(d+1)}$ for its first $d+1$ elements,
- vertices $v_{2neg_1}, v_{2neg_2}, \ldots, v_{2neg_{d+1}}$,
- for each variable x_i in the 3SAT instance, vertices v_{2x_i} and $v_{2\bar{x}_i}$.

We now turn to the description of edges in E. Figure 3.1 shows the set of gadgets that we are going to use. First, recall that, by definition, any solution to Bipartite-Mark(d) will put a $+1$ alteration on vertices $v_{21}, v_{22}, \ldots, v_{2(d+1)}$. We obtain a set of useful -1 alterations by considering the Minus gadget. In order to guarantee distortion at most d on vertex v_{1i}, that is, $|\Delta(v_{1i})| \leq d$, the only possible marks δ have to set a -1 alteration on vertex v_{2neg_i}. We copy this Minus gadget for vertices $(v_{11}, v_{2neg_1}), \ldots, (v_{1(d+1)}, v_{2neg_{d+1}})$ in order to enforce $(d+1)$ alterations -1 on vertices $v_{2neg_1}, \ldots, v_{2neg_{d+1}}$.

Second, we encode each boolean variable x_i with Variable gadgets. The first two gadgets above ensures that vertices v_{2x} and $v_{2\bar{x}}$ can only support a 0 or $+1$ alteration (-1 is not possible because, due to v_{2neg_i}, the distortion would be smaller than $-d$). The last two Variable gadgets below ensures that the sum of alterations on v_{2x} and $v_{2\bar{x}}$, that is $\delta(v_{2x}) + \delta(v_{2\bar{x}})$, is equal to 1 (if $d = 1$, edges from $v_{1x''}$ to v_{2i} are omitted). As a conclusion, any valid mark has to choose between v_{2x} or $v_{2\bar{x}}$ for a $+1$ alteration, and is forced to set a zero alteration on the remaining vertex. This mimics the boolean valuation of variable x.

It remains to encode clauses. The Clause gadget of Figure 3.1 encodes for example clause $x_1 \vee \bar{x}_2 \vee x_3$. Each literal x or $\bar{x}$ is encoded by an edge to vertex v_{2x} or $v_{2\bar{x}}$. Because of the edges to $v_{2neg_1}, \ldots, v_{2neg_{d+1}}$, at least one vertex encoding literals has to bear a $+1$ alteration so that the global distortion is greater than $-(d+1)$.

This construction guarantees that a Bipartite-Mark(d) solution corresponds to a $\#(d+1)$Mark($\leq d$) solution, and reciprocally. For a solution to the 3SAT instance, it is sufficient to choose $\delta(v_{2x}) = +1$ for each variables positively valued, and $\delta(v_{2x}) = 0$ otherwise. All other alterations are determined by this initial choice and the graph constraints. Conversely, given a solution of the Bipartite-Mark(d) problem, a satisfying valuation for the 3SAT instance is obtained by valuating positively all variables x such that $\delta(v_{2x}) = +1$, and by valuating negatively the remaining variables.

Finally, the encoding of N clauses on n variables requires $d^2 + 4d + 6n + N + 3$ vertices and $d^2 + 3d + n(4d+6) + N(d+4) + 2$ edges. The overall size of the produced graph is then linear in n and N, and hence in the size of the 3SAT input instance.

As a conclusion, this reduction from 3SAT shows that searching a valid mark for distortion $d \geq 1$ is an NP-hard problem. It is also in NP by guessing a mark and checking conditions. Hence the problem is NP-complete. This result generalizes to the special case $d = 0$ with a similar construct. □

We can now prove the theorem.

PROOF. [of Theorem 21] By reduction from BIPARTITE-MARK(d). We see the input bipartite graph $G = (E, V_1, V_2)$ as a structure, and consider the trivial query $\psi(u, v) = E(u, v)$. We choose a weight function $\mathcal{W}$ such that $\forall v \in V_2, \mathcal{W}(v) = 0$. Observe that $W = V_2$. Deciding if this structure has a watermarked weight function $\mathcal{W}'$ with $d + 1$ alterations $+1$ on its first elements with global distortion less or equal to d is equivalent to the decision of BIPARTITE-MARK(d). This latter problem is NP-complete, by Lemma 23. Furthermore, the used reduction is parsimonious, that is, preserves the number of solutions. Hence, for any $d \in \mathbb{N}$, $\#(d+1)\text{MARK}(\leq d)$ is $\#P$-complete, since counting the number of solutions of a NP-complete problem through parsimonious reductions is $\#P$-complete. □

3.2.2 Impossibility results

The difficulty of finding distortions that preserve the weight of all sets $\psi(\bar{a}, \mathcal{G})$ is related to the intrication of these sets. A powerful tool to study this intrication is the Vapnik-Chervonenkis dimension. Let V be a set and $\mathcal{C}$ be a family of subsets of V. A set $U \subseteq V$ is *shattered* by $\mathcal{C}$ if $\mathcal{C} \cap U = 2^U$, where $\mathcal{C} \cap U = \{C \cap U | C \in \mathcal{C}\}$. The VC-dimension $VC(\mathcal{C})$ of $\mathcal{C}$ with respect to V is the maximum of the sizes of the shattered subsets of V, or ∞ if the maximum does not exist. For a formula $\psi(\bar{u}, \bar{v})$ and a structure $\mathcal{G}$, let $\mathcal{C}(\psi, \mathcal{G}) = \{\psi(\bar{a}, \mathcal{G}) | \bar{a} \in \mathcal{U}^r\}$, and $VC(\psi, \mathcal{G}) = VC(\mathcal{C}(\psi, \mathcal{G}))$. We say that ψ has *bounded VC-dimension* on a class of structures $\mathcal{K}$ if there exists $k \in \mathbb{N}$ such that, for all $\mathcal{G} \in \mathcal{K}$, $VC(\psi, \mathcal{G}) \leq k$. We say that the VC-dimension is maximal if, for all $\mathcal{G} \in \mathcal{K}$, $VC(\psi, \mathcal{G}) = |W^{\mathcal{G},\psi}|$, hence when the full set of active tuples is shattered. In this situation, guaranteed watermarking for arbitrary structures, preserving even trivial queries is impossible.

Theorem 24 *In the non-adversarial view model, a problem $(\mathcal{K}, \psi)$ with maximal VC-dimension does not possess a scalable watermarking protocol.*

PROOF.

Let $(\mathcal{K}, \psi)$ be a watermarking problem with maximal VC-dimension. With at most k distorted weights, one can produce at most $\sum_{i=1}^{k} \binom{|W|}{i} 2^i \leq (2|W|)^k$ different weighted structures, encoding at most $O(k \ln |W|)$ bits. Suppose that there exists a scalable watermarking protocol encoding $|W|^q$ bits for a given q with distortion d_0. Then it must use a mark δ with at least $h(|W|) = \frac{1}{2} \frac{|W|^q}{\ln |W|}$ distortions with the same sign, say $+1$. The function h is increasing with respect to $|W|$ (since $|W|^q > \ln |W|$), and there exists n_0 such that $h(n_0) > d_0$.

We now consider a structure $\mathcal{G}_{n_0}$ which universe has n_0 elements. The watermarking protocol must add distortion $+1$ to weights from a set of elements $\mathcal{P}$ with $|\mathcal{P}| = h(n_0) > d_0$. Since, by hypothesis, $VC(\psi, \mathcal{G}_{n_0}) = |W^{\mathcal{G}_{n_0},\psi}|$, there exists a subset S of $\mathcal{U}^s$ of size $|W^{\mathcal{G}_{n_0},\psi}|$ which is shattered by sets in $\mathcal{C}(\psi, \mathcal{G}_{n_0})$. But since sets in $\mathcal{C}(\psi, \mathcal{G}_{n_0})$ are all subsets of W, there exists only one possible S: $S = W$ (sets in $\mathcal{C}(\psi, \mathcal{G}_{n_0})$ can not shatter sets outside W). So the set W is shattered by results of queries, and there exists a tuple $\bar{a}$ such that $\mathcal{P} = \psi(\bar{a}, \mathcal{G})$. Hence distortion on $\psi(\bar{a}, \mathcal{G}_{n_0})$ is greater than d_0, which contradicts the hypothesis to have a scalable watermarking protocol. □

It is worth noting that this impossibility argument can be followed with even trivial queries:

Theorem 25 *In the non-adversarial view model, there exists an FO formula ψ and a class of structures that do not possess a scalable watermarking protocol preserving ψ.*

PROOF. It is sufficient to consider the formula $\psi(u, v) \equiv E(u, v)$ and the class of structures $\mathcal{G}_n$ with $2^n + n$ vertices, and the simple binary relation E that links the ith vertex of the first 2^n vertices to the ith subset W_i of the n last vertices. The set of active tuples is clearly shattered by ψ and Theorem 24 applies. □

Remark 26 *Unbounded VC-dimension is not sufficient: one can construct a class of structures $\mathcal{G}_n$ of size n where only half of the active tuples are shattered ($VC(\psi, \mathcal{G}_n) = |W|/2$), with a $(|W|/4, 0)$-watermarking protocol. Consider the class of structures $\mathcal{G}_n$ with $2^{n/2}+1+n$ vertices, and the simple binary relation E that links the ith vertex of the first $2^{n/2}$ vertices to the ith subset of the $n/2$ last vertices, and the $2^{n/2}+1$th vertex a to all of the n last vertices. The watermarking problem defined by the query $\psi(u, v) = E(u, v)$ has n active tuples and unbounded VC-dimension. The last $n/2$ vertices of the active tuples are involved only for query $E(a, \mathcal{G}_n)$. Putting balanced distortions $(+1, -1)$ or $(-1, +1)$ only on their $n/2$ weights gives a watermarking protocol encoding $n/4$ bits with distortion 0.*

3.2.3 Lower bound on distortion

The previous results shows that, as soon that all subsets of size d of W can be captured by ψ, there is no scalable watermarking protocol with error smaller than $d/2$. This yields the following lower bound:

Lemma 27 *For any $d \in \mathbb{N}$, for any class of structures $\mathcal{K}$ and any logic with equality, there exists a query ψ such that $(\mathcal{K}, \psi)$ does not possess a scalable watermarking protocol with error smaller than d.*

Proof. For $d = 1$, consider the following query:

$$\psi(u_1, u_2, v) \equiv (v = u_1) \vee (v = u_2).$$

On any class of structures, this query captures all subsets of active tuples of size 2. If a scalable watermarking protocol exists, for large enough structures it must use at least 2 weight alterations of the same sign, say $(+1)$. Hence there exists parameters (a_1, a_2) capturing these two tuples, and distortions exceeds $d = 1$, which contradicts the hypothesis. This argument generalizes for any value of d. □

3.3 Watermarking while preserving local queries

3.3.1 Locality of queries

The previous sections show that, on generic structures, even trivial queries do not possess a scalable watermarking protocol. In this section we limit the relationship between tuples in the structure, and use the locality of queries to provide a scalable watermarking protocol. Given a structure $\mathcal{G} = \langle \mathcal{U}, R_1, \ldots, R_t \rangle$, its *Gaifman graph* [33] is the new structure $\langle \mathcal{U}, E \rangle$, where $(a, b) \in E$ if and only if there is a relation R_i in $\mathcal{G}$ and a tuple $\bar{c}$ in R_i such that a and b appear in $\bar{c}$. The distance $d(a, b)$ between two elements a and b is the length of a shortest path between a and b in the Gaifman graph of $\mathcal{G}$. If no such path exists, $d(a, b) = \infty$. Given $a \in \mathcal{U}$, $\rho \in \mathbb{N}$, the ρ-sphere $S_\rho(a)$ is the set $\{b | d(a, b) \leq \rho\}$, and for a tuple $\bar{c}$, $S_\rho(\bar{c}) = \cup_{a \in \bar{c}} S_\rho(a)$. Given a tuple $\bar{c} = (c_1, \ldots, c_n)$, its ρ-neighborhood $N_\rho(\bar{c})$ is defined as the structure $\langle S_\rho(\bar{c}), R_1 \cap S_\rho(\bar{c})^{r_1}, \ldots, R_t \cap S_\rho(\bar{c})^{r_t}, c_1, \ldots, c_n \rangle$, where $\forall i$, R_i has arity r_i. Let $\approx$ denotes isomorphism of structures. We consider the equivalence relation $\approx_\rho$ on elements of a structure $\mathcal{G}$ where $\bar{a} \approx_\rho \bar{b}$ if and only if $N_\rho(\bar{a}) \approx N_\rho(\bar{b})$. Finally, let $ntp(\rho, \mathcal{G})$ be the number of equivalence classes of the relation $\approx_\rho$. We introduce the important notion of the locality rank of a query:

Definition 28 *Given a query $\psi(u_1, \ldots, u_r)$, its locality rank is a number $\rho \in \mathbb{N}$ such that, for every $\mathcal{G} \in STRUCT[\tau]$ and two r-ary tuples $\bar{a}_1$ and $\bar{a}_2$ of $\mathcal{G}$, $N_\rho(\bar{a}_1) \approx N_\rho(\bar{a}_2)$ implies $\mathcal{G} \models \psi(\bar{a}_1) \Leftrightarrow \mathcal{G} \models \psi(\bar{a}_2)$. If no such ρ exists, the locality rank of ψ is ∞. A query is local if it has a finite locality rank. A language is local if each of its queries is local.*

For example, Gaifman's theorem [33] states that every first-order (relational calculus) query is local. The locality rank of a formula ψ is basically exponential in the depth of quantifier nesting in ψ, *but does not depend on the size of $\mathcal{G}$*.

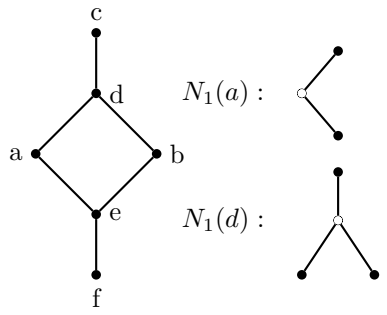


Figure 3.2: Instance and neighborhoods

Example 29 *We consider a graph instance $\mathcal{G} = \langle \mathcal{U}, R \rangle$ and the query $\psi(u, v) \equiv R(u, v)$ that enumerates all elements v at distance 1 of element u. This query has locality rank 1: it is sufficient to look at a neighborhood of radius 1 around u and v to devise if $\mathcal{G} \models \psi(u, v)$. Figure 3.2 shows $\mathcal{G}$ and neighborhoods $N_1(a)$ and $N_1(d)$ of elements a and d. Observe that there are 3 distinct neighborhoods of radius 1 (up to isomorphism), and that $N_1(a) \approx N_1(b), N_1(d) \approx N_1(e)$ and $N_1(c) \approx N_1(f)$.*

3.3.2 Watermarking and locality

In the sequel, we restrict our attention to structures in $STRUCT_k[\tau]$: structures with Gaifman graph of bounded degree k. An example of such structures is the class of graphs of bounded degree. Our aim is now to prove the following result:

Theorem 30 *In the non-adversarial view model, there exists a scalable watermarking protocol preserving any local query $\psi(u_1, \ldots, u_r, v_1, \ldots, v_s)$ on $STRUCT_k[\tau]$, with capacity $\Omega(|W|)$ and constant global distortion r.*

Proof. Let ψ be a formula of locality rank ρ, and $\langle \mathcal{U}, E \rangle$ be the Gaifman graph of $\mathcal{G}$. Let $type(\bar{b}) \in \mathbb{N}$ be the isomorphism type of any element $\bar{b} \in W$ with respect to relation $\approx_\rho$. We obtain a set $\mathcal{P}$ of watermarking positions by the following algorithm. Let t_0 be the isomorphism type of an equivalence class of $\approx_\rho$ on W with maximal size (that is, $\forall t \neq t_0, |\{\bar{b} | type(\bar{b}) = t\}| \leq |\{\bar{b} | type(\bar{b}) = t_0\}|$). Since there is a maximal number K of such isomorphism types, depending only on k, then $|\{\bar{b} | type(\bar{b}) = t_0\}| \geq |W|/K$. We build the set $\mathcal{P}$ of watermarking positions by repeatedly choosing an element $\bar{b}$ in $\{\bar{b} | type(\bar{b}) = t_0\}$ and by removing all elements in its 2ρ-neighborhood. Since a 2ρ-neighborhood has at most $sk^{2\rho}$ elements, the obtained set $\mathcal{P}$ has size at least $|W|/Ksk^{2\rho}$.

Then, we construct marks $\delta_{\mathcal{P}}$ according to the set $\mathcal{P}$ by assigning a zero distortion on tuples outside $\mathcal{P}$, and by assigning $+1$ or -1 distortions on tuples in $\mathcal{P}$, such that they sum up to 0:

$$\sum_{\bar{b} \in \mathcal{P}} \delta_{\mathcal{P}}(\bar{b}) = 0, \text{ and}, \forall \bar{b} \notin \mathcal{P}, \delta_{\mathcal{P}}(\bar{b}) = 0.$$

There is $2^{\Omega(|W|)}$ possible such marks $\delta_{\mathcal{P}}$. We now show that for any tuple $\bar{a}$, distortion $\Delta(\psi(\bar{a}, \mathcal{G})) = |\mathcal{W}'(\psi(\bar{a}, \mathcal{G})) - \mathcal{W}(\psi(\bar{a}, \mathcal{G}))|$ is a constant. First, observe that

$$\psi(\bar{a}, \mathcal{G}) = (\psi(\bar{a}, \mathcal{G}) \cap \mathcal{P}) \biguplus (\psi(\bar{a}, \mathcal{G})/\mathcal{P}),$$

where $\biguplus$ denotes disjoint union. Since $\delta_{\mathcal{P}}(\bar{b}) = 0$ whenever $\bar{b} \notin \mathcal{P}$, we have $\mathcal{W}'(\bar{b}) = \mathcal{W}(\bar{b})$ and

$$\Delta(\psi(\bar{a}, \mathcal{G})) = \Delta(\psi(\bar{a}, \mathcal{G}) \cap \mathcal{P}).$$

Second, for $B = S_\rho(\bar{a})^s$, the set of s-tuples in the neighborhood of tuple $\bar{a}$, we have

$$\Delta(\psi(\bar{a}, \mathcal{G})) \leq \Delta(\psi(\bar{a}, \mathcal{G}) \cap (\mathcal{P} \cap B)) + \Delta(\psi(\bar{a}, \mathcal{G}) \cap (\mathcal{P}/B)).$$

Because of the distance between $\bar{a}$ and elements in $\mathcal{P}$, the maximal size of $B \cap \mathcal{P}$ is r (this occurs when each element $a_1, \ldots, a_r$ of tuple $\bar{a}$ is in the neighborhood of an element in $\mathcal{P}$).

Notice also that for $\bar{b}_i, \bar{b}_j \in \mathcal{P}/B$, because of the choice of $\mathcal{P}$ and because $\bar{b}_i$ and $\bar{b}_j$ are not in the neighborhood of $\bar{a}$, $(\bar{a}, \bar{b}_i)$ and $(\bar{a}, \bar{b}_j)$ are isomorphic. Hence, due to the locality of ψ, $\mathcal{G} \models \psi(\bar{a}, \bar{b}_i) \leftrightarrow \mathcal{G} \models \psi(\bar{a}, \bar{b}_j)$. Then, by the definition of the watermarking method,

$$\sum_{\bar{b} \in \mathcal{P}} \delta_{\mathcal{P}}(\bar{b}) = 0.$$

Then

$$\sum_{\bar{b} \in \mathcal{P}} \Delta(\psi(\bar{a}, \bar{b})) = 0,$$

and finally

$$\sum_{\bar{b} \in \mathcal{P}/B} \Delta(\psi(\bar{a}, \bar{b})) \leq |B| \leq r.$$

The overall distortion $\Delta(\psi(\bar{a}, \mathcal{G}))$ is then smaller than r.

The scheme is deployed deterministically as follows. Given $(\mathcal{G}, \mathcal{W})$ and ψ, a precomputation phase, done once, generates the set $\mathcal{P}$. Given a word m of length l to hide, the marker returns $(\mathcal{G}, \mathcal{W}_m)$, where $\mathcal{W}_m$ is the m^{th} weights distortion of tuples in $\mathcal{P}$.

Once a suspect data server is localized, the detector (acting as a final user), asks the data server for all possibles queries, and obtains weights described in $\mathcal{W}^*$. By comparing them with the original ones, the corresponding subset of $\mathcal{W}^*$ is identified and message m is recovered.

The marker performs $O(ntp(\rho, \mathcal{G})|\mathcal{U}^r|)$ isomorphism tests on constant size graphs. The detector checks $O(|W|)$ values by querying the suspect server. Notice that the marker needs only to compute $\mathcal{P}$ once, and can compute from it every watermarked instance. □

Combined with the lower bound of Lemma 27, there exist queries with r parameters with watermarking error at least r. Hence the previous result is tight.

3.4 Preserving MSO-queries on trees and tree-like structures

3.4.1 Trees and automaton-definable queries

In this section we consider the problem of watermarking labeled trees and tree-like structures, while preserving MSO-queries. These structures can easily model XML documents.

Example 31 *Figure 3.3 shows an* XML *document with one of its possible* 1*-local distortion. We consider the following parametric XPATH query:*

$$\psi(a, v) = \texttt{/school/student[name=}a\texttt{]/exam},$$

that specifies the set of exam results of persons whose first name is a*. The goal is to watermark the document while preserving the sum of these exam results. This sum is* $\mathcal{W}(\psi(\texttt{Robert}, \mathcal{G})) = 28$ *on the original document, and has global distortion 2 on the second.*

XML deals actually with unranked trees, but several methods exist to encode them into binary trees (as in [78]), so we will restrict our attention to the binary case. We will use [37] notion of definability of a k-ary formula by a tree-automaton.

A binary tree is viewed as a $\{S_1, S_2 \preceq\}$-structure, where S_1, S_2 and $\preceq$ are binary relation symbols. A tree $\mathcal{T} = \langle T, S_1^{\mathcal{T}}, S_2^{\mathcal{T}}, \preceq^{\mathcal{T}} \rangle$ has a set of nodes T, a left child relation $S_1^{\mathcal{T}}$ and right child relation $S_2^{\mathcal{T}}$. Relation $\preceq^{\mathcal{T}}$

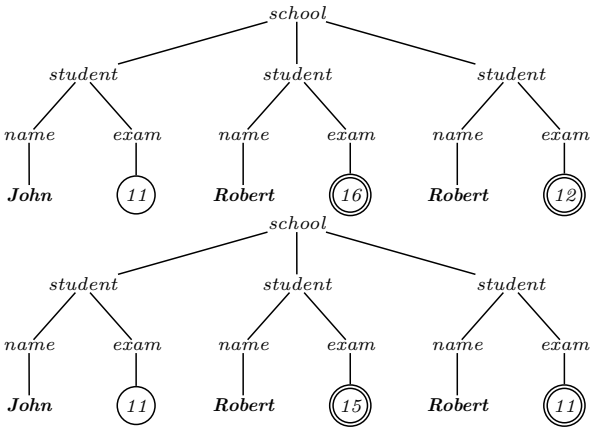


Figure 3.3: An XML documents and a possible 1-local distortion

stands for the transitive closure of $S_1^{\mathcal{T}} \cup S_2^{\mathcal{T}}$, the tree-order relation. Given a subset U of nodes in T, we denote by $lca(U)$ the least common ancestor of all elements in U according to order $\preceq^{\mathcal{T}}$. A weighted tree $(\mathcal{T}, \mathcal{W})$ is a tree with a weight assignment $\mathcal{W} : T^s \to \mathbb{N}$. Given a finite alphabet Σ, let $\tau(\Sigma) = \{S_1, S_2, \preceq\} \cup \{P_c | c \in \Sigma\}$ where for all $c \in \Sigma$, P_c is a unary symbol. A Σ-tree is a structure $\mathcal{T} = \langle T, S_1^{\mathcal{T}}, S_2^{\mathcal{T}}, \preceq^{\mathcal{T}}, (P_c^{\mathcal{T}})_{c\in\Sigma}\rangle$, where its restriction $\langle T, S_1^{\mathcal{T}}, S_2^{\mathcal{T}}, \preceq^{\mathcal{T}}\rangle$ is an ordered binary tree and for each $a \in T$ there exists exactly one $c \in \Sigma$ such that $a \in P_c^{\mathcal{T}}$. We denote this unique c by $\sigma^{\mathcal{T}}(a)$.

We consider trees with a finite number of distinguishable *pebbles* placed on vertices. For some $k \geq 1$, let $\Sigma_k = \Sigma \times \{0, 1\}^k$. For a Σ-tree $\mathcal{T}$ and a tuple $\bar{a} = (a_1, \ldots, a_k)$ of vertices of $\mathcal{T}$, let $\mathcal{T}_{\bar{a}}$ be the Σ_k-tree with the same underlying tree as $\mathcal{T}$ and $\sigma^{\mathcal{T}_{\bar{a}}}(b) = (\sigma^{\mathcal{T}}(b), \alpha_1, \ldots, \alpha_k)$, where $\alpha_i = 1$ if and only if $b = a_i$.

A Σ-tree automaton is a tuple $\mathcal{B} = (Q, \delta, F)$. Set Q is a set of states, and $F \subseteq Q$ is a set of accepting states. Function $\delta : ((Q \cup \{*\})^2 \times \Sigma) \to Q$ is the transition function ($* \notin Q$). A run $\rho : T \to Q$ of $\mathcal{B}$ on a Σ-tree $\mathcal{T}$ is defined as follows. If a is a leaf then $\rho(a) = \delta(*, *, \sigma^{\mathcal{T}}(a))$. If a has two children b_1 and b_2, then $\rho(a) = \delta(\rho(b_1), \rho(b_2), \sigma^{\mathcal{T}}(a))$. If a has only a left child b then $\rho(a) = \delta(\rho(b), *, \sigma^{\mathcal{T}}(a))$ and similarly if a has only a right child b, $\rho(a) = \delta(*, \rho(b), \sigma^{\mathcal{T}}(a))$. Finally, a Σ_{k+s}-tree automaton defines a s-ary query with k parameters $\mathcal{B}(\bar{a}, \mathcal{T}) = \{\bar{b} \in T^s | \mathcal{B} \text{ accepts } \mathcal{T}_{\bar{a}\bar{b}}\}$ on each Σ-tree $\mathcal{T}$. Let $W = \bigcup_{\bar{a}} \mathcal{B}(\bar{a}, \mathcal{T})$.

It is well known that MSO-sentences and tree-automata have the same expressive power. For formula with free variables, a Σ_k-tree automaton is equivalent to an MSO-formula $\psi(u_1, \ldots, u_k)$ of vocabulary $\tau(\Sigma)$ if for all Σ-tree, $\mathcal{B}(\mathcal{T}) = \psi(\mathcal{T})$.

Lemma 32 ([37]) *For any MSO-formula $\psi(u_1, \ldots, u_k)$ of vocabulary $\tau(\Sigma)$ there exists a Σ_k-tree automaton $\mathcal{B}$ that is equivalent to ψ.*

3.4.2 Preserving MSO-queries

Our final goal is now to prove the following theorem:

Theorem 33 *In the non-adversarial view model, there exists a scalable watermarking protocol preserving any MSO-definable query on trees or classes of structures with bounded clique-width or bounded tree-width.*

To prove this result, we first prove the following theorem, in order to apply Lemma 32.

Theorem 34 *In the non-adversarial view model, given a query ψ defined by a tree automaton with m states, there exists a scalable watermarking protocol preserving ψ with capacity $\frac{|W|}{4m}$.*

We begin by the following lemma:

Lemma 35 *Let $\mathcal{B}$ be a Σ_2-tree automaton with m states. Then for every Σ-tree $\mathcal{T}$, there exists $n = |W|/4m$ distinct sets $V_1, \ldots, V_n \subseteq W$ and n distinct pairs $(b_i, b'_i) \in {V_i}^2$ of distinct elements such that $\forall i \neq j, V_i \cap V_j = \emptyset$, and $\forall a \in T$:*

$$a \notin V_i \rightarrow (b_i \in \mathcal{B}(a, \mathcal{T}) \leftrightarrow b'_i \in \mathcal{B}(a, \mathcal{T})).$$

Proof. Informally, we will iterate a construct from [37]: from the bottom-up, we form $|W|/4m$ subtrees of $\mathcal{T}$ of size at least $2m$. Since the automaton has only m states, one can find in each V_i a pair of vertices such that the automaton ends in the same state on a given subtree, for all $a \notin V_i$. Then, the automaton can not distinguish between these elements. By using a pair marking on such pairs, the overall distortion is controlled.

More formally, from the bottom-up of $\mathcal{T}$, let U_1 be a minimal subtree with respect to inclusion with at least $2m$ elements. Since $\mathcal{T}$ is binary, U_1 contains at most $4m$ elements. We can repeat this construct $2n = \lfloor |T|/4m \rfloor$ times, obtaining sets $U_1, \ldots, U_{2n}$.

We consider the binary relation F on $H = \{U_1, \ldots, U_{2n}\}$ to be the set of all pairs (U_i, U_j) such that $lca(U_i) \prec^{\mathcal{T}} lca(U_j)$, and there is no k such that $lca(U_i) \prec^{\mathcal{T}} lca(U_k) \prec^{\mathcal{T}} lca(U_j)$. Then (H, F) is a forest with $2n$ vertices and at most $2n - 1$ edges. Therefore there is at most n elements of this forest with more that 1 child. Without loss of generality, suppose that $U_1, \ldots, U_n$ have at most one child.

If U_i has no child, let $V_i = \{v \in T | lca(U_i) \preceq^{\mathcal{T}} v\}$, the elements of the subtree of $\mathcal{T}$ rooted at $lca(U_i)$. If U_i has one child U_j, then let $V_i = \{v \in T | lca(U_i) \preceq^{\mathcal{T}} v$ and $lca(U_j) \not\preceq^{\mathcal{T}} v\}$ the set of all vertices of the subtree of $\mathcal{T}$ rooted at $lca(U_i)$ that are not in the subtree rooted at $lca(U_j)$. Observe that $V_1, \ldots, V_n$ are pairwise disjoint.

Let $1 \leq i \leq n$. If U_i has no child, since $|U_i| \geq 2m$ is greater than the number m of automaton states, there exists two distinct elements $b_i, b'_i \in U_i$ such that:

For all $a \notin V_i$, automaton $\mathcal{B}$ running on $\mathcal{T}_{ab_i}$ or $\mathcal{T}_{ab'_i}$ reaches $lca(U_i)$ in state q_i.

Now if U_i has a child U_j, and $q_1, \ldots, q_m$ are the states of $\mathcal{B}$, we define pairs $b_{i,k}, b'_{i,k}$ for $1 \leq k \leq m$ by induction on k. Suppose $1 \leq k \leq m$ and that $b_{i,l}$ and $b'_{i,l}$ are already defined for $l < k$. Since $|U_i| \geq 2m$ we have $|U_i \backslash \{b_{i,1}, \ldots, b_{i,k-1}\}| > m$. Therefore there exists distinct elements $b_{i,k}, b'_{i,k} \in U_i \backslash \{b_{i,1}, \ldots, b_{i,k-1}\}$ such that:

There is a state $q_{i,k}$ of $\mathcal{B}$ such that if $a \notin V_i$, the automaton running on either $\mathcal{T}_{ab_i}$ or $\mathcal{T}_{ab'_i}$ and leaving $lca(U_j)$ in state q_k reaches $lca(U_i)$ in state $q_{i,k}$.

Finally, if U_i has no child, and $a \notin V_i$, $\mathcal{B}$ accepts $\mathcal{T}_{ab_i}$ if and only if $\mathcal{B}$ accepts $\mathcal{T}_{ab'_i}$. If U_i has one child U_j, $a \notin U_i$ and $\mathcal{B}$ ends in $lca(U_j)$ in state q_t, $\mathcal{B}$ accepts $\mathcal{T}_{ab_{i,t}}$ if and only if $\mathcal{B}$ accepts $\mathcal{T}_{ab'_{i,t}}$. □

We now claim that pairs (b_i, b'_i) are good candidates for a watermarking algorithm.

Proof. [of theorem 34] Let $(b_1, b'_1), \ldots, (b_n, b'_n)$ be the n distinct pairs of Lemma 35, W' be a subset of $\{1, \ldots, n\}$, and consider a mark that applies $(+1, -1)$ or $(-1, +1)$ distortions on pairs (b_i, b'_i). There are 2^n such marks that sums all to zero. For a Σ_2-tree automaton, let $a \in T$. Suppose there is a j such that $a \in V_j$. Notice that in this case j is unique. Then for all $i \neq j$, $a \notin V_i$ and $\mathcal{B}$ accepts $\mathcal{T}_{ab_i}$ if and only if it accepts $\mathcal{T}_{ab'_i}$. Since distortion on weights b_i and b'_i is zero, distortion on $\mathcal{W}(\mathcal{B}(a, \mathcal{T}))$ is limited by the pair b_j, b'_j. This distortion is at most 1. Otherwise, if $\forall i, a \notin V_i$, then the induced distortion of all pairs is 0. Finally, the number of pairs is $|W|/4m$. This result generalizes to a Σ_{r+s}-tree automaton with distortion at most r.

The time complexity of this method corresponds to a full traversal of the tree $\mathcal{T}$ to form subsets of size at most $4m$, and for each set, to a search of pairs of nodes that force the automaton to end in the same state. It takes at most $|T|$ steps to visit the tree and $(4m)^2$ runs of the automaton on $|T|/4m$ sets. The overall complexity is then $O(m|T|)$. □

We can now end with the proof of the main theorem.

Proof. [of theorem 33]

We prove the existence of a scalable watermarking protocol preserving MSO-formulas on (1) tree structures, (2) bounded clique-width structures and (3) bounded tree-width structures:

1. Let ψ be an MSO-formula on trees. By applying Lemma 32, we obtain an automaton $\mathcal{B}$ equivalent to ψ. Then, by Theorem 34, there is a corresponding scalable watermarking protocol preserving $\mathcal{B}$, hence ψ, on trees.

2. For structures with bounded clique-width, we apply Lemma 16 of [37]: given a structure $\mathcal{G}$ with bounded clique-width, we can construct a labeled *parse-tree* $\mathcal{T}$ such that, for any MSO-formula $\psi(\bar{u})$ there exists a MSO-formula $\tilde{\psi}(\bar{u})$ such that $\psi(\mathcal{G}) = \tilde{\psi}(\mathcal{T})$. This reduce the bounded clique-width case to the tree case and, by step (1) of this proof, there exists a scalable watermarking protocol preserving $\tilde{\psi}$, hence ψ, on bounded clique-width structures.

3. Finally, structures with bounded tree-width k have a bounded clique-width of at most 2^k. Then, by step (2) of this proof, there exists a scalable watermarking protocol preserving MSO-formulas on bounded tree-width structures.

□

If we lower the restriction on the class of structures, scalability is lost:

Theorem 36 *In the non-adversarial view model, there exists an MSO formula ψ and a class of structures with unbounded tree-width that do not possess a scalable watermarking protocol preserving ψ.*

Proof.

Example 19 in [37] exhibits a MSO formula ψ with unbounded VC-dimension on the class of *grids*, which has unbounded tree-width. This shows actually that for all grid $\mathcal{G}$, the set $\bigcup_{\bar{a}} \psi(\bar{a}, \mathcal{G})$ is shattered by sets in $\{\psi(\bar{a}, \mathcal{G}) : \bar{a} \in \mathcal{U}^r\}$. The corresponding watermarking problem with the same formula ψ on the same class of structures is such that for all $\mathcal{G}, W^{\mathcal{G},\psi} = \bigcup_{\bar{a}} \psi(\bar{a}, \mathcal{G})$. Hence for all $\mathcal{G}$, its set of active tuples is shattered, so $VC(\psi, \mathcal{G}) = |W^{\mathcal{G},\psi}|$, showing by Theorem 24 that no scalable watermarking protocol is possible. □

3.5 Adversarial model

Up to now, we considered a non-adversarial model. In the *adversarial model*, data servers can perform any reasonable distortion on the watermarked structure $\mathcal{G}_m$, or on their query answers. The reasonable limit of the global distortion is denoted d'. Once data servers are allowed to perform distortions, deterministic detectors are no longer possible: as a matter of fact, an unstoppable strategy for the attacker is to guess the inserted mark and its position, and to modify weights accordingly. Hence we consider detectors with a controlled probability of error δ, where δ is chosen by the data owner (with for example $\delta = 10^{-6}$).

More formally, a probabilistic algorithm has the ability to pick a random bit b at each step, and to adapt its computation according to the value of b. Hence a given computation is a path in the tree of all possible random choices along with its corresponding probability: both form a probability space Ω. We consider a watermarking protocol that may succeed with high probability and may fail (stop and abandon), or produce an incorrect result with a small probability:

Definition 37 *Given $0 \leq \delta < 1$ and $l, d, d' \in \mathbb{N}$, a (l, d, d', δ)-watermarking protocol preserving ψ in the adversarial view model is a (l, d)-watermarking protocol preserving ψ such that the detector, using only answers $\mathcal{A}(\psi, \mathcal{G}^*)$ from a d'-global distortion $\mathcal{G}^*$ of $\mathcal{G}_m$, verifies:*

$$\Pr_{\Omega}[\mathcal{D} \text{ outputs } m] \geq 1 - \delta.$$

[61] proposed a general technique to turn a non-adversarial protocol into an adversarial one. Their result is quite general as it applies to their edge model (that corresponds to our view model), and also to their distance model, that corresponds to a detector using only the aggregate value $\mathcal{W}(\psi(\bar{a}, \mathcal{G}))$ of the query answer, and not all the set $\mathcal{A}(\psi(\bar{a}, \mathcal{G}))$ (we will elaborate on this distinction in the conclusion section). We recall their result here for completeness, adapted to our view model, and refer the reader to the original paper for a more precise exposition.

Two natural hypothesis are used to constraint the behavior of the attacker. Let $(\mathcal{G}, \mathcal{W})$ be an original structure, $(\mathcal{G}, \mathcal{W}')$ be a watermarked version, and $(\mathcal{G}, \mathcal{W}^*)$ be a voluntary alteration of $(\mathcal{G}, \mathcal{W}')$. The first assumption indicates that there is a limit to the distortion the attacker can add to a structure, imposed by its intended use:

Assumption 38 (Bounded distortion) *The attacker respects the global distortion assumption, for an absolute constant d', that is, for all $\bar{a} \in \mathcal{U}^s$,*

$$|\mathcal{W}^*(\psi(\bar{a}, \mathcal{G})) - \mathcal{W}(\psi(\bar{a}, \mathcal{G}))| \leq d'.$$

It is noteworthy that there is no relationship between d' and the owner maximal distortion d. This allows a large freedom to the attacker.

Let S be the set of marks used by the marker algorithm, and Q be the set of tuples queried by the detector. The second assumption guarantees the probabilistic independency between the set of queries Q and the set of marks S. Given a mark $\delta : W \to \mathbb{Z}$, we define $\mathcal{W}_\delta$ such that, for all $\bar{b} \in W$, $\mathcal{W}_\delta(\bar{b}) = \mathcal{W}(\bar{b}) + \delta(\bar{b})$.

Definition 39 *A set $Q \subseteq W$ is low-bias with respect to $S \subseteq \{-1, 0, +1\}^W$ if for all $\bar{a} \in Q$,*

$$|\delta(\bar{a})| \leq 1 \text{ and } \forall z \in \{-1, 0, +1\}, \Pr_{\delta \in S}[\delta(\bar{a}) = z] \leq \frac{1}{2}.$$

Definition 40 *Let $\gamma, p > 0$. A set $S \subseteq \{-1, 0, +1\}^W$ is (γ, p)-unpredictable if for any $Q \subseteq W$ such that Q is low-bias with respect to S and $|Q| = \omega(1)$, any strategy $\mathcal{W}^*$ available to the adversary satisfies*

$$\Pr_{\delta \in S}\left[\sum_{\bar{a} \in Q} [\mathcal{W}^*(\bar{a}) = \delta(\bar{a})] > (\frac{1}{2} + \gamma)|Q|\right] < p.$$

Assumption 41 (Limited knowledge) *The attacker has limited knowledge on the mark distribution of the owner, that is, for any $S \subseteq \{-1, 0, 1\}^W$ such that $|S| = \omega(1)$, S is (γ, p)-unpredictable.*

This second assumption indicates that the attacker does not know exactly what information has been introduced into the structure (and does not know the original, non-marked structure). This models also the situation where a server is indeed not malicious, but uses data from an other source, similar to the owner's database (false positive detection).

Khanna and Zane's framework is the following. Given a watermarking protocol in the non-adversarial model for $\psi(u_1, \ldots, v_1, \ldots, v_s)$, the framework requires a pair of distinct values $v_1, v_2 \in \{-1, 0, 1\}$, a set of marks $S \subseteq \{-1, 0, 1\}^{|W|}$, and a set of query parameters $Q = \{\bar{a}_1, \ldots, \bar{a}_L\} \subseteq \mathcal{U}^s$ such that:

- Q and S have a significant size, that is $|Q|, |S| \geq \omega(1)$;
- Q is low bias with respect to S;
- The allowed global distortion is d, that is, for all $\delta \in S$, for all $\bar{a} \in \mathcal{U}^s$

$$|\mathcal{W}_\delta(\psi(\bar{a}, \mathcal{G})) - \mathcal{W}(\psi(\bar{a}, \mathcal{G}))| \leq d.$$

- Any set of query answers can be mapped to a unique $\delta \in S$: for any $D : Q \to \{v_1, v_2\}$, there is a unique $\delta \in S$ such that

$$\forall \bar{a} \in Q, \mathcal{W}_\delta(\bar{a}) = \mathcal{W}(\bar{a}) + D(\bar{a}).$$

Theorem 42 *[61] Under the Bounded distortion and Limited knowledge assumptions, given $\gamma < \frac{1}{9(d'+1)}$ and $p \geq e^{-o(L)}$, any non-adversarial scalable watermarking protocol consistent with the framework can be used to encode $O(L)$ bits in the adversarial model, with a detector's error probability at most* $\max\{2p, o(1)\}$.

Observe that the watermarking robustness is obtained by lack of knowledge (an attacker knows there is a mark, but do not know its amplitude and distribution) and not by using the intractability of a computational problem, like in the cryptographic setting. All the watermarking protocols presented in this chapter comply with Khanna and Zane's framework, so extend to the adversarial setting.

Corollary 43 *There exists a scalable watermarking protocol in the* adversarial *view model preserving*

- *local queries on structures with bounded degree Gaifman graph;*
- *MSO-queries on trees and structures with bounded clique-width or tree-width.*

Proof.

By Theorem 30 and 33, there exists a non-adversarial scalable protocol for the above logical fragments and classes of structures. Let $d \in \mathbb{N}$ be the allowed global distortion.

For local languages, by the proof of Theorem 30, there is a subset $\mathcal{P} \subseteq W$ such that any distortions δ restricted on $\mathcal{P}$ summing up to 0 lead to a correct structure with global distortion smaller than d. We split $\mathcal{P}$ into disjoint pairs $V_1, \ldots, V_n$. For any pair $V_i = \{\bar{b}_{i1}, \bar{b}_{i2}\}$, the set of marks S is obtained by choosing uniformly and independently between $\{\delta(\bar{b}_{i1}) = +1, \delta(\bar{b}_{i2}) = -1\}$ or $\{\delta(\bar{b}_{i1}) = 0, \delta(\bar{b}_{i2}) = 0\}$, for each set V_i. Any such choice also leads to an overall distortion on $\mathcal{P}$ smaller than d. For MSO-queries, Lemma 35 also provides such sets $\mathcal{P}$ and $V_1, \ldots, V_n$, suited for the same set of marks S. Let $v_1 = +1$ and $v_2 = 0$. We choose the set of query $Q = \{\bar{b}_{11}, \ldots, \bar{b}_{1n}\}$, that is the first tuple of each pair.

Because the considered protocols are scalable, $|\mathcal{P}| = 2n$ is $\Omega(|W|)$, hence $|Q|$ and $|S|$ are $\omega(1)$, and moreover $|S|$ is exponential in $|Q|$. Q is low-bias with respect to S because alterations in S are chosen independently and uniformly. The maximum distortion implied by any δ is smaller than the prescribed d. For any valuation $D : Q \to \{v_1, v_2\} = \{1, 0\}$ of recovered values from Q by the detector, we can find the unique corresponding mark δ:

$$\forall \bar{b}_{i1} \in Q, \delta(\bar{b}_{i1}) = D(\bar{b}_{i1}), \delta(\bar{b}_{i2}) = -D(\bar{b}_{i1}).$$

This protocol corresponds to the framework. Hence, by Theorem 42, $O(n)$ bits can be encoded such that the error probability of the detector is at most $\max\{2p, o(1)\}$. Finally, $n = \lceil |W|/2 \rceil = \Omega(|W|)$. Thus, there exists an adversarial scalable protocol for the considered logical fragments and classes of structures. □

We do not consider here the general problem of *collusion attacks*, where servers combine several watermarked copies of the database to erase the watermark (a full exposition is given in [67]).

3.6 Instance model and non-parametric queries

In this section we consider two easier models that guarantee a huge watermarking bandwidth.

Instance model In this model, the detector as full access to the suspect instance: the marker can hide information outside the set of active tuples W. Let $\psi(u_1, \ldots, u_r, v_1, \ldots, v_s)$ be the query to be preserved. The corresponding watermarking bandwidth then depends on the size of W, relatively to the size of the set $\mathcal{U}^s$ of all possible tuples:

Theorem 44 *In the instance model, there exists a* $(|\mathcal{U}^s| - |W|, 0)$*-watermarking protocol preserving any query* $\psi(u_1, \ldots, u_r, v_1, \ldots, v_s)$*.*

Proof. Let $F = \mathcal{U}^s \backslash W$. Hence, any tuple in F does not participate in any set $\psi(\bar{a}, \mathcal{G})$, thus the corresponding weights have no impact on any $\mathcal{W}(\psi(\bar{a}, \mathcal{G}))$. By putting a distortion 0 or $(+1)$ on each tuple in F, $|F|$ bits can be hidden with global distortion 0. The time complexity related to this protocol corresponds to the computation of the set of active tuples W. □

If $|\mathcal{U}^s| - |W|$ is small, that is if W covers a non-negligible part of $\mathcal{U}^s$, the problem reduces to the view model.

Non-parametric queries Queries considered up to now have parameters fulfilled by data users. In the present setting, users simply obtain the result of a set of predefined queries without parameters $\psi_1(v_1, \ldots, v_s)$, $\ldots, \psi_k(v_1, \ldots, v_s)$ (observe that there is no parameter $\bar{u}$):

Theorem 45 *There exists a $(|W|, 0)$-watermarking protocol preserving non-parametric queries $\psi_1(v_1, \ldots, v_s)$, $\ldots, \psi_k(v_1, \ldots, v_s)$.*

PROOF. Let $\bar{w} \in W$ be an active tuple. Being active, it belongs to the image of some ψ_i on $\mathcal{G}$. Let $cl(\bar{w}) = \{i | \bar{w} \in \psi_i(\mathcal{G})\}$ be the class of $\bar{w}$, that is the set of query indices whose image contains $\bar{w}$. Observe that two tuples of the same class appear in the same set of queries. Hence, if their alterations sum up to zero, the overall distortion on their common set of queries is also zero. There is at most 2^k distinct classes. Hence, there is set $W_0 \subseteq W$ of size at least $|W|/2^k$ such that $\forall \bar{w}, \bar{w}' \in W_0, cl(\bar{w}) = cl(\bar{w}')$. Consider any partition of this set into pairs, and any mark that sums up to zero on each pair. This yields $2^{|W_0|}$ different marks. The partition size is $|W|/2^{k+1}$, which is $\Omega(|W|)$ (this bound can be enhanced to $|W| - 2^k$ by considering all possible classes, and discarding at most 2^k tuples because of odd-size classes). Computing the active tuples W and their respective class can be done in one pass, and the overall time complexity of this protocol is equivalent to the computation of the set W. □

3.7 Incremental watermarking

In this section we suppose that a data owner needs to update the database and propagates changes to each of the registered data servers. The problem is then to maintain the watermark he has inserted. In the sequel, an update u is a function that maps a weighted structure $(\mathcal{G}, \mathcal{W})$ to an updated structure $u((\mathcal{G}, \mathcal{W}))$. Hence, a sequence of updates $u_1, \ldots, u_n$ yields a sequence of weighted structures $(\mathcal{G}^1, \mathcal{W}^1) = u_1((\mathcal{G}^0, \mathcal{W}^0)), \ldots, (\mathcal{G}^n, \mathcal{W}^n) = u_n((\mathcal{G}^{n-1}, \mathcal{W}^{n-1}))$.

Definition 46 *For a class of updates U, a watermarking protocol maintaining U is a triplet $(\mathcal{M}, \mathcal{M}_U, \mathcal{D})$, where:*

- *$\mathcal{M}$ is a marker as in Definition 14, producing, given an original structure $(\mathcal{G}^0, \mathcal{W}^0)$ and an initial word m, a watermarked structure $(\mathcal{G}^0_m, \mathcal{W}^0_m)$;*
- *$\mathcal{M}_U$ is a function that, given an update $u \in U$ on the owner's current structure $(\mathcal{G}^i, \mathcal{W}^i)$, outputs a new watermarked structure $(\mathcal{G}^i_m, \mathcal{W}^i_m)$;*
- *$\mathcal{D}$ is a detector as in Definition 14, but for any watermarked structure $(\mathcal{G}^i_m, \mathcal{W}^i_m)$.*

Let $(\mathcal{G}, \mathcal{W})$ be a weighted instance. We consider *weights-only updates*: a weights-only update $u = (\bar{a}, \Delta)$ where $\bar{a} \in \mathcal{U}^s, \Delta \in \mathbb{N}$, is an update that alters only the weight $\bar{a}$ by the value Δ, and leaves the finite part unchanged, that is:

- $u((\mathcal{G}, \mathcal{W})) = (\mathcal{G}, \mathcal{W}')$;
- $\mathcal{W}'(\bar{a}) = \mathcal{W}(\bar{a}) + \Delta$, and $\forall \bar{a}' \neq \bar{a}, \mathcal{W}'(\bar{a}) = \mathcal{W}(\bar{a})$.

For this class, updating the watermarked instance is easy.

Theorem 47 *Previous watermarking protocols maintain weights-only updates.*

PROOF. First, recall that the set of active tuples, W, is not sensitive to the weights values, as its definition rely only on the finite structure $\mathcal{G}$. Then tuples involved in queries do not change during an update. Let δ be the used mark. For a mark distortion

$$\mathcal{W}^i_m(\bar{a}) = \mathcal{W}^i(\bar{a}) + \delta(\bar{a}),$$

in the current watermarked instance $(\mathcal{G}_m^i, \mathcal{W}_m^i)$, and a weights-only update $(\bar{a}, \Delta)$ such that

$$\mathcal{W}^{i+1}(\bar{a}) = \mathcal{W}^i(\bar{a}) + \Delta,$$

we propagate the *same update* Δ:

$$\mathcal{W}_m^{i+1}(\bar{a}) = \mathcal{W}_m^i(\bar{a}) + \Delta.$$

The same global distortion is obtained for the new instance. Since the detector extracts the watermark by computing the difference between the watermarked and original weights, for example,

$$\mathcal{W}_m^{i+1}(\bar{a}) - \mathcal{W}^{i+1}(\bar{a}) = (\mathcal{W}^i(\bar{a}) + \Delta + \delta(\bar{a})) - (\mathcal{W}^i(\bar{a}) + \Delta) = \delta(\bar{a}),$$

it is only sensitive to the watermark δ, and not the update Δ. Hence, any weights-only update $(\bar{a}, \Delta)$ on the original structure is simply propagated as an update $(\bar{a}, \Delta)$ on the watermarked one. □

3.8 Practical aspects

The proposed protocols are rather efficient: first, the set W of active tuples has to be computed. Then, a set of watermarking positions $\mathcal{P}$ has to be computed once, and the proper watermarking and detection operations takes linear time in $|\mathcal{P}|$. Finding the set $\mathcal{P}$ has a linear time complexity according to the size of the structure $\mathcal{G}$: for example, the marker for local languages on structures with bounded-degree Gaifman graph performs $O(ntp(\rho, \mathcal{G})|\mathcal{U}^r|)$ isomorphism tests on constant size graphs. The marker for an MSO query ψ on trees requires $O(m|T|)$ steps, where m is the size of the automaton simulating ψ. It is known [107] that in the worst case, m is in $tower(n)$, where n is the size of the formula ψ. The *tower* function is such that $tower(0) = 1$ and for $i \geq 1, tower(i) = 2^{tower(i-1)}$. The main difficulty from the computational point of view is then the role of these huge constants.

Similarly, while error bounds presented in this work elaborate on previous ones, the watermarking rate suffers from the involved constants. Indeed, the watermarking rate is $|W|/Ksk^\rho$, where K is the number of possible isomorphism type of connected graphs with at most k vertices, and ρ is the locality rank of the query, which is exponential in the number of quantifier alternation of ψ. But the method of compensating distortions is still fruitful in the practical context. As shown in [67], a heuristic search of pairs is likely to succeed, yielding only a minute distortion. In this latter work, several practical questions not considered in the present work were resolved. We mention them here for completeness:

- Identification of tuples weights, using key dependencies (in the present protocol, the owner has to specify its instance as a finite part and a weighted part, which is not always trivial);
- Blindness of the protocol: the present protocol required the original data set for detection, which is not always affordable. The $(+1, -1)$ pairing was enhanced to an exchange of distinct bits between two values;
- Robustness against collusion attack: malevolent data servers may compare their watermarked data set in order to locate differences. By choosing marks in a collusion-secure fingerprinting code [110], robustness against a prescribed collusion size in guaranteed.

3.9 Conclusion

In this chapter we considered the problem of watermarking databases or XML documents, while preserving a set of queries in a specified language $\mathcal{L}$. We gave structural arguments for the existence of a watermarking protocol related to the VC-dimension of sets definable in $\mathcal{L}$. We showed that watermarking on arbitrary instances is impossible, and that languages and structures with bounded VC-dimension established by Grohe and Turán have also good watermarking properties. But we do not know if bounded VC-dimension is a sufficient condition to obtain a scalable watermarking protocol.

This work can be extended in various directions. First, one can elaborate on the error model: we considered an absolute error on local and global distortions, while classical studies from the approximation literature consider relative errors rather than absolute ones, because relative approximation is preserved under composition. Observe however that a relative perturbation $1 \pm \varepsilon$ of weights always yields a global distortion of at most $1 + \varepsilon$. Hence the watermarking problem becomes trivial. But a mix between local absolute error and global relative error can be considered.

Second, our model is limited to the instance and view models. The adversarial framework from [61] adapts also to a more general *aggregate view model*, where only the numerical results of queries is needed to perform detection. We think that, by controlling the impact of alterations on queries for each parameter, protocols for this aggregate model could be obtained for the same logical fragments and classes of structures we considered here.

Third, our model does not capture exactly the result from [61] since shortest path queries are indeed an *optimization problem* (notice however that the VC-dimension of weighted graphs with respect to their shortest path is bounded). Optimization has received a large interest from the finite model theory community [62,86]. An interesting point is to find relationships between logical definability of such problems, mainly their weighted versions [122], and their watermarking capacity.

Related publications

- **David Gross-Amblard. Query-Preserving Watermarking of Relational Databases and XML Documents. To appear in *ACM Transactions on Database Systems (ACM TODS)*, 36(1), 2010 (tentatively scheduled).**
- **David Gross-Amblard. Query-Preserving Watermarking of Relational Databases and XML Documents. In *ACM Principles of Database Systems (PODS)*, 2003, pages 191–201.**

4

Practical aspects: the WATERMILL system

In this section we elaborate on the theoretical results of the preceding chapter to convey to a practical solution of query-preserving watermarking. We also discuss other resolution methods like linear programming.

Our contribution In this section, we present WATERMILL [27], an optimized watermarking and fingerprinting system for relational databases. It features a built-in *usability constraints definition language* as well as an efficient watermarking engine to override the limitations of the greedy method. Our system achieves the following capabilities:

- *Speeding-up the watermarking/fingerprinting process.* We identify a set of constraints patterns for which it is possible to translate the watermarking problem into an integer linear program (ILP). Valid watermarks are found among the solutions of this system. These patterns capture what we call *weight-independent constraints*, that include aggregate and join computations, which are central in the design of usability constraints used in real-world datasets [77]. To produce valid watermarks, we propose two approaches, both implemented in WATERMILL. The first one uses existing ILP solvers. The second one, the *Pairing* algorithm, searches for pairs of compensating alterations. It performs a precomputation to obtain a simple description of a huge set of valid watermarks. After precomputation, finding several valid watermarks is immediate (linear time). This second approach is far much faster and best suits huge databases (millions of tuples).
- *Resisting attacks from a collusion of purchasers, while preserving usability*: when distributing several fingerprinted versions of a database, the owner is exposed to collusion attacks. In this setting, several malicious purchasers may collude to compare their watermarked versions. By locating positions where their documents differ, they can discover where alterations were performed. By modifying documents on these positions, they might obtain a new version without a readable watermark, hence evading detection. The design of efficient collusion-secure fingerprinting codes is a long-standing effort [17, 47, 110]. To achieve collusion-security, watermark messages must be carefully chosen from a precise *codebook*, and inserted in the *same* positions in all the distributed versions. We show that for the aforementioned weight-independent constraints, a family of good watermarks can be quickly found, so that bit alterations are always performed at the same positions. Hence using a collusion-secure code is possible, while preserving usability constraints.

4.1 Databases watermarking

4.1.1 Example and Hypothesis

Example The example database `mills` shown in Figure 4.1 is used throughout the chapter. It consists of a single relation, `mills`, containing descriptions of powerplants around towns in France. Each powerplant is characterized by its `type`, its coordinates `(x,y)`, the `place` where it is located, its `height` and its `production`.

mills instance – mills relation

x	y	place	type	prod	height
30	53	Dinan	windmill	125	60
62	56	Challans	windmill	223	90
55	22	Colmar	mill	443	5
22	51	Chalain	mill	53	2
50	18	Dijon	geothermic	33	15

Figure 4.1: The original `mills` instance

mills2 instance – mills relation

x	y	place	type	prod	height
35	53	Dinan	windmill	127	60
62	56	Challans	windmill	203	91
55	22	Colmar	mill	463	5
22	51	Chalain	mill	53	2
50	18	Dijon	geothermic	33	15

mills3 instance – mills relation

x	y	place	type	prod	height
40	53	Dinan	windmill	125	60
62	56	Challans	windmill	203	90
55	22	Colmar	mill	423	5
22	51	Chalain	mill	63	2
50	18	Dijon	geothermic	33	15

Figure 4.2: Two fingerprinted instances

Note that the `place` attribute is the primary key of the relation `mills`. In Figure 4.2 are shown two examples of fingerprinted `mills`, namely `mills2` and `mills3`. They differ from the original data on several positions, *e.g.*, `prod` and `height` of `place Challans`. Remark that only integer values have been modified and that primary keys have not been altered. When a suspect copy is discovered, the owner extracts his watermark. If this watermark is similar to the one inserted in `mills2` or `mills3` (in a way defined later), he can claim ownership over the dataset and charge the corresponding purchaser with illegal redistribution. We call *marker* the algorithm for watermark insertion, and *detector* the algorithm for watermark detection.

Hypothesis We use the following common assumptions [7, 71–73]:

- *Numerical values*: We suppose that databases contain at least one column of numerical values, preferably integers. Watermark insertion is performed by modifying least significant bits of these numerical values.
- *Primary keys*: We suppose that every relation has a primary key column and that this column, even if numerical, is not altered during the watermarking process, or by the attacker. This assumption is justified by the fact that (*i*) primary keys are the articulations between the different relations of the database: for an attacker, it is possible but not easy to rename them in an uninvertible manner; and (*ii*) primary keys often have a public meaning. For example, `place Dinan` can not be changed to `Paris` without misleading all the users of a stolen dataset, hence annihilating all the efforts of the attacker.

Adversarial model In a naïve setting, the stolen database is kept identical to the purchased one. In a more realistic, *adversarial setting*, a malicious purchaser might alter the stolen dataset (up to a realistic extent) in order to erase the watermark. Classic types of attack are:

- *Random data alteration*: a subset of data is randomly distorted. The amplitude of this alteration is limited so that the dataset remains valuable;

- *Data loss*: a subset of data is discarded;
- *Mix-and-match*: new data from an unknown dataset with the same schema is added to the dataset.

A good watermarking system is robust against such alterations.

4.1.2 Watermarking Basics

Watermarking identified values The main hypothesis, used also in [7, 39, 103], is that modified values (*i.e.* watermark positions) must be *identified values*, *i.e.* must be in the scope of a primary key. Consequently modified values are clearly identified, which helps watermark recovery in the adversarial setting. This is one of the main differences with classic multimedia watermarking, for which similar keys do not exist.

Let d be a database. A numerical value appearing in the database is said to be an *identified value* if this precise value can be uniquely identified in the entire dataset by its relation name r, attribute name a and primary key value p . The tuple (r, a, p) is called the *value identifier*. In our example database `mills`, the value identifier $i_0 = (mills, height, Challans)$ denotes the `height` of the windmill of `Challans`. When the relation name is clear from the context, we use the notation a_p for value identifiers (*e.g.* $height_{Challans}$). Clearly, if all relations possess a primary key, every numerical value is an identified one. We denote by $\mathcal{I}(d)$ the set of all value identifiers of database d. For example, $\mathcal{I}(mills)$ contains all `x`, `y`, `height` and `prod` identifiers for key values in $\{Dinan, Challans, Colmar, Chalain, Dijon\}$. Finally, the set of all identified values of the database d is seen as a vector $\vec{v}(d)$ indexed by value identifiers from $\mathcal{I}(d)$. On our example databases, the height of the windmill of Challans, denoted by i_0 is $\vec{v}(mills)[i_0] = 90$ on database $mills$, and $\vec{v}(mills2)[i_0] = 91$ on database $mills2$.

Watermarking method (starting point) We recall here Agrawal, Haas and Kiernan's method [7] for watermarking relational databases. It is used as a reference for the subsequent algorithms. Note also that a different method, from Sion et al. [103] can also be used, but we do not consider it here due to lack of space. The method relies on an integer pseudo-random generator $\mathcal{S}$ whose draws can not be predicted without the knowledge of its *seed*. Several parameters are used: the secret key $\mathcal{K}$, the ratio $1/\gamma$ of watermarked elements, the maximum number ξ of alterable least significant bits (LSB), and the maximum probability δ of detection errors[1]. The algorithm respects the following steps: for each value identifier i, the random number generator is seeded with $\mathcal{K}$ concatenated with i ($\mathcal{K}$ and i are seen as binary strings). If the first integer produced by $\mathcal{S}$ is 0 modulo γ, then the value $\vec{v}(d)[i]$ is considered for watermarking. In the binary expression of $\vec{v}(d)[i]$, a position is chosen according to the next integer from $\mathcal{S}$, computed modulo ξ. The bit located at this position in $\vec{v}(d)[i]$ is replaced by a mark bit, whose value is given by the parity of the third production of $\mathcal{S}$. The detection algorithm proceeds identically by locating bit positions in a suspect dataset. The binary values found at these positions are then compared with the expected ones, and the number of correct matches is recorded. If the match ratio exceeds a given threshold, the dataset is declared suspect.

This method exhibits several important properties: *robustness, accuracy, incremental, public system and blindness* (see [7] for a complete discussion). Among them, blindness means that the original dataset is not needed for detection: only the watermarking parameters $(\mathcal{K}, \gamma, \xi, \delta)$ and the suspect dataset are required. Blindness is crucial in the context of very large datasets, since their backup, copy or transmission to a trusted authority might not be possible.

In the sequel, we show how to add usability constraint preservation to this basic building block.

4.2 A Declarative Language for Usability Constraints

A watermarked version of a database is obtained by changing the values of some attributes. Consequently the results of some queries on the watermarked dataset are likely to be modified. The purchaser or the owner of the watermarked dataset might want to limit the impact of these modifications, both on attribute

[1] The number ν of relational attributes available is also used, but our use of value identifiers already captures the attribute selection.

values and on query results, by specifying some *usability constraints*. Similarly to [39, 103], we distinguish between the set $\mathcal{L}$ of *local usability constraints*, that affect tuples individually, and the set $\mathcal{G}$ of *global usability constraints*, that apply on a subset of tuples.

4.2.1 Constraints Examples

We introduce here the syntax of our declarative language through a list of short examples. The owner might want to enforce the quality of the watermarked dataset by defining a set of local constraints $\mathcal{L} = \{C_1, C_2, C_3\}$, asking for a maximum alteration of 10 units on map coordinates x and y (C_1), 1 meter on height of a windmill (C_2), and 20KW on a production measure (C_3). Documents `mills2` and `mills3` of Fig. 4.2 respect each constraint in $\mathcal{L}$.

```
local 10 on mills.x, mills.y          # C1
local 1  on (select height from mills# C2
             where type=windmill)
local 20 on mills.prod                # C3
```

A tuple value e is a *modifiable value* of a dataset d if e is within the scope of at least one local constraint in $\mathcal{L}$. In this work, alterations are done only on values that are *both identified and modifiable*. We denote by $\mathcal{M}(d)$ the set of these elements. By definition, $\mathcal{M}(d) \subseteq \mathcal{I}(d)$. We now enrich our example with a set of global constraints $\mathcal{G} = \{C_4, C_5, C_6, C_7, C_8, C_9\}$. We say that a watermark is *valid* if it simultaneously respects $\mathcal{L}$ and $\mathcal{G}$. Constraint C_4 allows a maximal variation of 10 for the total production of mills and windmills.

```
global 10 on (                        # C4
  select sum(prod) from mills
  where type in (mill,windmill))
```

Constraint C_5 expresses that, for legal reasons, productions under 60 KW must remain under this limit *after* watermarking.

```
invariant (select place from mills  # C5
           where prod < 60)
```

Document `mills2` respects global constraints C_4 and C_5, but not document `mills3` (overall production for windmills and mills increased by more than 10 units and production for `Chalain` exceeds 60KW). Constraint C_6 imposes that the distance between the Dinan powerplant and a power collector located in position $(10, 10)$ on the map must not be distorted by more than 5 units.

```
global 5 on (                         # C6
  select sqrt((x-10)^2+(y-10)^2)
  from mills where place=Dinan)
```

Constraint C_7 states that powerplants with equal heights should still have equal heights after watermarking (but not necessarily with the same value). Note that this kind of pattern is a convenient way to express foreign key constraints between tables.

```
invariant(                            # C7
  select m1.place, m2.place from mills m1,m2
  where  m1.height = m2.height)
```

When constraints do not fit the previous patterns, call to an external checking program (named *e.g.* `qualityChecker`) can also be used.

```
call "qualityChecker"                 # C8
```

Finally, arithmetic expressions can be used. Suppose that heights are expressed in *feet* for a windmill, and in *meter* for a mill. Constraint C_9 states that, for administrative reasons, the sum of the heights of the powerplants of `Challans` and `Colmar` must not exceed a given limit, expressed in meters.

```
invariant (                          # C9
  select 0.304*p1.height + p2.height < 30
  from mills as p1,mills as p2
  where p1.place=Challans and p2.place=Dinan)
```

4.2.2 Semantics

We now give the formal semantics of these constraints (this section can be skipped in a first reading). In the sequel, $\varphi(d)$ denotes the result obtained by applying an SQL query φ on document d. The identified part $\mathcal{I}(\varphi, d)$ is a subset of the set of identified values $\mathcal{I}(d)$. An identified value $(r, a, p) \in \mathcal{I}(\varphi, d)$ if and only if its value is in $\varphi(d)$ and can be traced through φ. For instance, for the query `select type from mills where height>80`, $\varphi(d)$=`{windmill}`. Although two tuples are of type `windmill`, only the one identified by place `Challans` participates to the result; hence, $\mathcal{I}(\varphi, d) = \{(mills, type, Challans)\}$. Computing $\mathcal{I}(\varphi, d)$ is not always possible [18]. In what follows, we use only queries φ for which the computation of $\mathcal{I}(\varphi, d)$ is straightforward. It is the case for our previous sample queries. A conditional sentence φ is a conjunction/disjunction combination of terms $R.A\ \theta\ c$ where R is a relation, A an attribute, $\theta \in \{<, =, >\}$ and c a constant. A set of identifiers $\mathcal{I}$ is said to be *independent* of φ if no identifier in $\mathcal{I}$ has a relation name and an attribute name of one of the terms of φ.

- Constraint `local` p `on` ψ: where $p \in \mathbb{N}$ and ψ is a query identifying single attributes. For k local constraints with queries $\psi_1, \ldots, \psi_k$, the set of *modifiable values* $\mathcal{M}(d)$ is the set of all identified elements of $\psi_1, \ldots, \psi_k$, *i.e.* $\mathcal{M}(d) = \bigcup \mathcal{I}(\psi_j, d)$. Remark that constraints `C1` and `C3` are written in a convenient abbreviated form whose expansion into the usual syntax is obvious. For instance $C3$ is expanded into `local 20 on (select prod from mills)`. A watermarked dataset $d_{\vec{w}}$ is such that, for all $i \in \mathcal{M}(d)$, $\vec{v}(d_{\vec{w}})[i] = \vec{v}(d)[i] + \vec{w}[i]$, *i.e.* each value $\vec{v}(d_{\vec{w}})[i]$ in the watermarked dataset is the sum of the original value $\vec{v}(d)[i]$ and an alteration $\vec{w}[i]$. A dataset $d_{\vec{w}}$ is said to respect the local constraint if and only if $\forall i \in \mathcal{I}(\psi, d), |\vec{w}[i]| \leq p$.
- Constraint `global` p `on` ψ: where $p \in \mathbb{N}$ and ψ is a query returning a numerical value. A dataset $d_{\vec{w}}$ is said to respect the global constraint if and only if $|\psi(d) - \psi(d_{\vec{w}})| \leq p$.
- Constraint `invariant` (ψ): a watermarked dataset $d_{\vec{w}}$ satisfies this constraint if $\psi(d) = \psi(d_{\vec{w}})$.
- Constraint `call(program)`: represents a call to an external program that checks constraints (*e.g.* a computation not easily definable in SQL). This clause is respected by dataset $d_{\vec{w}}$ if the *program* answers "yes" with d and $d_{\vec{w}}$ as input. This corresponds to the usability plugins of [103].

Finding alterations of the dataset that respect such constraints may be a difficult computational task. The next section shows how we optimize the discovery of such alterations for specific patterns of constraints.

4.3 Fingerprinting as an optimization problem: ILP reduction

4.3.1 Splitting Constraints

Most of the previous constraints can be translated into linear constraints, i.e. inequations on sum of values from the dataset. Our approach is then to split the set $\mathcal{G}$ of usability constraints into two sets: the set *Lin* of linear constraints, and the set of remaining constraints *Gen* that can not be translated (e.g. `call` constraints). We will resolve usability constraints from *Lin* using an ILP solver, obtaining a partial instantiation of a good watermark vector, say $\vec{w}_1$. Watermark positions left undefined are denoted by $\mathcal{M}/\vec{w}_1$. The remaining

constraints from *Gen* are explored using the greedy method `GreedyMark` [103] on positions $\mathcal{M}/\vec{w}_1$, obtaining a complete watermark vector $\vec{w}$. Next sections present the automatic translation of constraints and our watermarking algorithm.

4.3.2 Translation into Linear Constraints

Recalling the previous example, translation of constraint C_9 is immediate:

$$0.304 * \vec{w}[\texttt{height}_{\texttt{Challans}}] + \vec{w}[\texttt{height}_{\texttt{Colmar}}] \leq 30. \quad \#\texttt{C9}$$

Constraints C_1 to C_5 can also be expressed by means of linear inequalities, e.g.:

$$-10 \leq \vec{w}[\texttt{x}_{\texttt{Dinan}}] \leq 10. \quad \#\texttt{C1}$$

Constraints C_6 and C_8 can not be linearized: there is no reason for C_8 to be linear and C_6 is quadratic. Observe also that C_7 can be linearized (as explained in the sequel), but its conditions do not hold in the original dataset. Based on the previous example, we identify a set of constraint patterns that can be directly translated into a linear program $\mathcal{P}$. These patterns express useful usability constraints on the data and can be easily recognized. We consider four patterns: local constraints and weight-independent [invariant/join/sum] constraints. For each pattern, we give its general syntactic form, specific restrictions that must be checked, and its translation into a linear inequation.

1. Local constraints (*e.g.* C_1, C_2, C_3):
   - Pattern: `local` p `on` ψ;
   - Restrictions: ψ respects the following pattern:

     `select ... from ... where` φ`,`

     where φ is an SQL condition *independent* of $\mathcal{M}(d)$;
   - ILP constraint: $\forall i \in \mathcal{I}(\psi, d), -p \leq \vec{w}[i] \leq p$.
2. Weight-independent sum constraints (wis-constraints): a constraint is said to be *weight-independent* if the set of value identifiers involved in the query computation is the same, whatever the perturbations on identified values are. For instance, C_4 is weight-independent: even if `prod` is modified, the set of identified values in `windmill` or `mill` that are involved in the computation of C_4 does not change. This property allows to compute once for all the set of identified values used in a query computation, and to assign variables to these values in the linear system. An example of weight-*dependent* constraint is given below:

   ```
   global 10 on (select sum(prod) from mills where prod<100).
   ```

   If a `prod` is equal to 99, watermarking it to 100 will exclude it of the previous linear encoding. A sufficient condition to obtain the weight-independence property is that the where clause is followed by an SQL condition *independent* of $\mathcal{M}(d)$. The formal pattern is the following:
   - Pattern: `global` p `on` ψ;
   - Restrictions: ψ has the following pattern:

     `select sum(attName) from relName where` φ`,`

     where φ is an SQL condition *independent* of $\mathcal{M}(d)$.;
   - ILP constraint: $-p \leq \sum_{i \in \mathcal{I}(\psi', d)} \vec{w}[i] \leq p$, with ψ'=`select attName from relName where` φ.

The weight-independent sum pattern can be easily extended to handle the `mean` aggregate. However, quadratic constraints, *e.g.* on the standard deviation, can not be expressed.

3. Weight-independent invariant constraints (for example C_5):
   - Pattern: `invariant`(ψ);
   - Restriction: ψ has the following pattern:

     `select ... from ... where` φ `and` $A\ \theta\ c$,

     where φ is an SQL condition *independent* of $\mathcal{M}(d)$, A is an attribute, $\theta \in \{=, <, >\}$ and $c \in \mathbb{N}$;
   - ILP constraint: for all $i \in \mathcal{I}(\psi_A, d)$:

     $$\vec{w}[i] + \vec{v}(d)[i]\ \theta\ c,$$

     with $\psi_A =$`select A from ... where` φ
4. Weight-independent join constraints (for example C_7):
   - Pattern: `invariant`(ψ);
   - Restriction: ψ has the following pattern:

     `select ... from ... where` φ `and` $A = B$,

     where φ is an SQL condition *independent* of $\mathcal{M}(d)$, A, B are attributes;
   - ILP constraint: for any pair of value identifiers $(i, j) \in \mathcal{I}(\psi_{AB}, d)$ with ψ_{AB} defined by

     `select A,B from ...where` φ `and A=B`,

     we add constraint $\vec{w}[i] = \vec{w}[j]$.

4.3.3 Algorithm

Clearly, any watermark satisfying the linear system respects all *Lin* constraints. Our aim is then to extend Agrawal, Haas and Kiernan's algorithm [7] so that only good watermarks are selected. The sketch of the resulting algorithm is as follows:

1. we compute the distortions using the method of Agrawal, Haas and Kiernan and we memorize them in a vector $\vec{\Delta}$ ($\vec{\Delta}[i]$ is the distortion on identified value i);
2. we create a new (0-1) variable $\vec{s}[i]$;
3. for each identified value i, we add the constraint $\vec{w}[i] = \vec{s}[i].\vec{\Delta}[i]$ to the linear program obtained by translating the usability constraints;
4. the watermark is the solution of the above integer linear program $\mathcal{P}$ that maximizes the number of values $\vec{s}[i]$ which are equal to 1.

The overall algorithm includes Li, Swarup and Jajodia's extension of the initial Agrawal, Haas and Kiernan's algorithm [7] to fingerprinting [73] (see Alg. 1). In order to hide the binary message m, at each watermarking step, a bit of m is pseudo-randomly chosen. This bit is masked by an exclusive OR with a pseudo-random bit. Symbol $S_t(k)$ denotes the output number t of a pseudo-random generator initialized with the seed k (see [7]). For the watermark detection we use the values at positions i such that $\vec{s}[i] = 1$ (see Alg. 2) to recover the message. The value of each recovered bit is chosen using a majority voting. This procedure is called `ThresholdMajority` (omitted due to space limitations) and returns the word formed by the bits whose vote values exceed a predefined threshold $1/2 + \alpha$ ($0 < \alpha < 1/2$). Remark that some bits of the word might remain undefined, *e.g.* when there are 50% of votes for both values 0 and 1.

4.3.4 Properties

Blindness Our algorithm is blind in the sense of [103], *i.e* it does not require the original dataset for detection. However, as explained in [103], positions used for a constraint-preserving watermarking *must* be recorded for future detection (in our case, the positions i where $\vec{s}[i] = 1$). Indeed, the recomputation of the vector $\vec{s}$ on the watermarked dataset as the solution of the linear program does not necessarily yields to the same value as the one computed before watermarking. Having to backup this set is not a real limitation since it can be efficiently compressed (*e.g.* by simple interval encoding).

Robustness Note that when no usability constraints are to be preserved, this algorithm yields exactly the same watermark as the one proposed by Agrawal, Haas and Kiernan (*i.e.* when all $\vec{s}[i]$ are equal to 1). Therefore, its robustness against attacks is the same. The robustness of the algorithm varies when usability constraints are taken into account. The more restrictive the constraints are, the easier it is to guess the location of watermarked bits. A too complex group of constraints may even yield a non-watermarkable dataset. Nonetheless experimental evaluations show that we are still able to find watermarks on practical constraints (see Section 4.7).

Algorithm 1: `LinearMark` (dataset d, message m, local constraints $\mathcal{L}$, linear constraints Lin, arbitrary constraints Gen, parameters $\mathcal{K}, \xi, \gamma$)

1 $\mathcal{P} \leftarrow \emptyset$; `/* empty linear program */`
2 $\mathcal{M}(d) \leftarrow$ `ExtractModifiableIdentifiers` $(d,\mathcal{L})$;
3 **foreach** *modifiable identifier* $i \in \mathcal{M}(d)$ **do**
4 **if** $(S_1(i \circ \mathcal{K}) \mod \gamma = 0)$ **then** `/* try mark this element */`
5 $j \leftarrow S_2(i \circ \mathcal{K}) \mod \xi$ `/* bit index */`
6 $k \leftarrow S_3(i \circ \mathcal{K}) \mod |m|$; `/* letter index */`
7 $mask \leftarrow S_4(i \circ \mathcal{K}) \mod 2$;
8 $mark \leftarrow m[k] \oplus mask$; `/* mark bit */`
9 $ovalue \leftarrow \vec{v}[i]$;
10 $mvalue \leftarrow \vec{v}[i]$;
11 $mvalue[j] \leftarrow mark$;
 `// compute distortion` $\vec{\Delta}[i] \in \{-2^j, 0, 2^j\}$
12 $\vec{\Delta}[i] \leftarrow (ovalue - mvalue)$;
13 Add linear constraints to $\mathcal{P}$ $\begin{cases} 0 \leq \vec{s}[i] \leq 1 \text{ and } \vec{s}[i] \in \mathbb{N}; \\ \vec{w}[i] \leftarrow \vec{s}[i].\vec{\Delta}[i]; \end{cases}$
14 Add translation of constraints from Lin to $\mathcal{P}$;
15 $\vec{w}_1 \leftarrow$`LinearSolve` $(\mathcal{P}, \max_{\vec{s}}(\#s_i = 1))$;
16 $\vec{w} \leftarrow$`GreedyMark` $(m,\mathcal{M}/\vec{w}_1,\mathcal{K},\mathcal{L},Gen)$;
17 **return** $(\vec{v} + \vec{w}, \vec{s})$;

Problem reduction State-of-the-art ILP solvers (like Ilog Cplex,Dash Xpress-Mp, IBM OSL, etc.) can handle classically up to 10^4 variables. If the number of modifiable values exceeds this limit, which is likely to occur on large datasets, several methods can be used: apply standard reduction techniques to lower the number of useful variables ([99, 117]), work only on active identifiers, *i.e.* those which are used in query evaluation, or choose a random subset of variables, or group them according to a secret key. In what follows, we present a heuristic which allows to compute efficiently a subset of solutions of the ILP.

Algorithm 2: `LinearDetect`(suspect dataset d,$\vec{s}$,message length l,key $\mathcal{K}$,ξ)

```
1  for k in {1,...,l} do
2  |  vote[k][0] ← 0 ;
3  |  vote[k][1] ← 0 ;
4  foreach identifier i in d appearing in s⃗ do
5  |  j ← S2(i ∘ K) mod ξ;                          /* bit index */
6  |  k ← S3(i ∘ K) mod l;                          /* letter index */
7  |  mask ← S4(i ∘ K) mod 2;                       /* mask bit */
8  |  readMark ← v⃗[i][j] ⊕ mask;                    /* read the mark */
9  |  vote[k][readMark]← vote[k][readMark] + 1;
   // voting
10 return ThresholdMajority (vote);
```

4.4 Fingerprinting as an optimization problem: Pairing Heuristic

4.4.1 Pairing Algorithm

For large datasets, using an ILP solver might not be efficient. So, we introduce a heuristic which allows to find a large number of valid watermarks, by analyzing the form of the usability constraints. This heuristic can be used for a subset of weight independent constraints, more specifically the subset of sum constraints. The result is a set of l watermark positions where *all* binary messages of size l can be encoded. Any message encoding on these positions will necessarily lead to valid watermarks w.r.t. usability constraints. The main point is that these positions are computed *once for all*. No other usability constraint checking is needed for further message encodings (for constraints respecting our patterns). We call this method the *pairing* algorithm since it uses pairs of compensating alterations. The difference with the previous algorithm is twofold. First, the pairing algorithm does not consider all the possible valid watermarks, but focuses only on a restricted family. Thus, fewer watermarking bits can be discovered. Nevertheless the computing time is smaller, since it does not solve an integer linear program. Second, its algorithm can be delegated to the RDBMS, allowing for a better scalability.

4.4.2 Pairing for Weight-independent Sum Constraints

Suppose that ψ is a weight-independent sum constraint. The set of tuples contributing to the sum are the same in the original instance and in the watermarked one. Hence, the distortion induced by the addition of a watermark $\vec{w}$ is only the sum of marks $\vec{w}[i]$ for $i \in \mathcal{I}(\psi, d)$. We illustrate the algorithm on our example, for the following weight-independent sum constraints:

$\psi 1 \equiv$ `global 0 on (`
`  select sum(prod) from mills where type=windmill),`
$\psi 2 \equiv$ `global 0 on (`
`  select sum(prod) from mills where type in (windmill,mill));`

In this setting, identified productions correspond to primary keys *Dinan*, *Challans*, *Colmar* and *Chalain*. Values associated with these primary keys are $125, 223, 443, 53$. Query ψ_1 has $125 + 223$ as a result, hence it depends on `prod` from *Dinan* and *Challans*. Query ψ_2 has $125 + 223 + 443 + 53$ as a result, and it depends on `prod` from *Dinan*, *Challans*, *Colmar* and *Chalain*. We represent this information in the following *dependency matrix* $A(\psi_1, \psi_2)$:

	Dinan	*Challans*	*Colmar*	*Chalain*
ψ_1	1	1	0	0
ψ_2	1	1	1	1

Let $i_1, \ldots, i_{2l}$ be the set of value identifiers. The aim of the pairing algorithm is to partition these values *into l dependency pairs* $\{(i_1^1, i_1^2), \ldots, (i_l^1, i_l^2))\}$, so that, for all $j \in \{1, \ldots, l\}$,

- i_j^1 and i_j^2 are involved in the *same constraints*;
- watermark distortions on i_j^1 and i_j^2 will be opposite.

Going back to our example, productions of *Dinan* and *Challans* are involved in $\{\psi_1, \psi_2\}$. Productions of *Colmar* and *Chalain* participate only to $\{\psi_2\}$. Hence the first pair will be $(Dinan, Challans)$ and the second $(Colmar, Chalain)$. For example, hiding message "10" could be achieved by applying the following alterations:

- $+1$ on the prod of *Dinan* and -1 on the prod of *Challans* for the first pair;
- -1 on the prod of *Colmar* and $+1$ on the prod of *Chalain* for the second pair.

Observe that the overall distortion on constraints ψ_1 and ψ_2 will always be 0.

Ensuring blindness In order to produce a blind algorithm, the alteration on a pair (i^1, i^2) will only depend on the primary key of i^1. According to this key and to the allowed local distortion, we secretly choose a bit position *index*. If the numerical values $\vec{v}[i^1]$ and $\vec{v}[i^2]$ are equal on this bit position, we can not use the pair for watermarking. On the contrary, if they differ on this position, we can permute these bits without altering the usability constraints. Our method is the following:

- We choose a pseudo-random binary mask *mask*;
- To encode a '1', we put $(1 \oplus mask)$ on $\vec{v}[i^1][index]$ and $(0 \oplus mask)$ on $\vec{v}[i^2][index]$;
- To encode a '0', we put $(0 \oplus mask)$ on $\vec{v}[i^1][index]$ and $(1 \oplus mask)$ on $\vec{v}[i^2][index]$.

Since these bits were different in the original dataset, this operation does not change their sum, and the contribution of this pair to the global distortion is still zero. The complete algorithm includes the precomputation phase (see Alg. 4) and the actual watermarking process (see Alg. 3). Note that, similarly to the previous query-preserving algorithms [103], the set of positions used for watermarking must be recorded for further detection. Anyway, this set allows for an efficient compact representation.

4.4.3 Detection

The detector considers each identified value $\vec{v}'[i]$ that was potentially watermarked, based on the set of recorded pairs and on the secret key. The pseudo-random generator is used afterwards, similarly to [7,73], to obtain the position j of the watermark bit. The bit value $\vec{v}'[i][j]$ on this position is masked by an exclusive OR with a pseudo-random mask bit. The result is stored in $readMark$. This bit accounts for the letter k of the hidden message m, where k is also computed by the pseudo-random generator. The vote for the value $readMark$ of this bit k is incremented in the *vote* array (see Alg. 5). Then, each position k in the detected message is assigned one of values 0,1 or *'undef'*. A binary value is assigned if the number of votes for one value is significantly higher than for the other one, using majority voting (`ThresholdMajority`).

4.4.4 Enhancements and Extension to other Constraints

`ComputePairs`, efficient pairs computation The core of the method is the computation of pairs of identifiers. Its implementation must be optimized to achieve scalability. In our prototype, this task is mainly devoted to the RDBMS by computing and storing the dependency matrix as a relation. Suppose that $\psi_1, .., \psi_n$ are n constraints. We create a relation `matrix:(id,dep_pattern)` such that the `id` column will contain the identifiers and the column `dep_pattern` their dependencies expressed by binary patterns. If (i, p) is a tuple from `matrix`, the k-th bit of p has the value 1 if ψ_k depends on the tuple identified by i,

and 0 otherwise. For our examples, the tuples of `matrix` are $(Dinan, 11), (Challans, 11), (Colmar, 01)$ and $(Chalain, 01)$. In order to compute the sets of identifiers which have the same dependencies, we order the tuples in `matrix` according to their binary pattern. Then the tuples involved in the same constraints are shuffled.

```
SELECT id FROM matrix
ORDER BY dep_pattern, md5(concat(Kp,id));
```

Pairs of identifiers having the same dependencies are obtained by reading pairs of identifiers from the sorted `matrix` relation.

Algorithm 3: `PairMark`(document d,message m, wis-constraints $\mathcal{C}$,$\mathcal{L}$,$\mathcal{K}$, ξ, γ)

```
1  pairs= ComputePairs (d,C,L,K);
   // pairs of equal classes, with a pseudo-random order
   // This computation is done only once
2  foreach pair of identifiers (i^1, i^2) in pairs do
       // Try to mark this pair
3      if (S_1(i^1 ∘ K) mod γ = 0) then
4          j ← S_2(i^1 ∘ K) mod ξ;                          /* bit index */
5          k ← S_3(i^1 ∘ K) mod |m|;                        /* letter index */
6          if (v[i^1][j] ≠ v[i^2][j]) then                  /* bits 1-0 or 0-1 */
7              mask ← S_4(i^1 ∘ K) mod 2;
8              mark ← m[k] ⊕ mask;                          /* mark bit */
9              v[i^1][j] ← mark;
10             v[i^2][j] ← not (mark);
11             if constraints in L are violated then
12                 v[i^1][j] ← not (mark);
13                 v[i^2][j] ← mark;                        /* undo modifications */
14             else
15                 add (i^1, i^2) to markList;
16
17
18 return markList;
```

Matrix reduction and non-zero constraints It is possible to reduce the number of lines of the dependency matrix. Observe first that if ψ_1 and ψ_2 depend on *exactly* the same values, we can use only ψ_1 in the dependency matrix without changing the solution. Second, this technique fits well for zero distortion constraints. For the sake of simplicity, suppose that we only encode the marks +1 or -1, and that all the constraints have the same global distortion t. Hence, if two queries ψ_1 and ψ_2 depend on the same values except on t positions, using only ψ_1 in the dependency matrix may introduce a maximal distortion of at most t on query ψ_2. For the general case, we delete all queries that are identical up to t divided by the maximal allowed local distortion on each element.

Capacity Theoretical arguments [39] show that we are likely to find pairs on natural datasets. We assess this property by our experiments in Section 4.7.

Handling join and invariant constraints For a join condition $\vec{w}[i] = \vec{w}[j]$, we suppress j from the set of modifiable identifiers $\mathcal{M}$. When a value is assigned to $\vec{w}[i]$, we propagate it to $\vec{w}[j]$. For invariant constraints, we simply set to zero the allowed distortion on each considered identified value.

Algorithm 4: `ComputePairs`(document d, wis-constraints $\{\psi_1, \ldots, \psi_k\}$,$\mathcal{L}$,$\mathcal{K}$)

```
1  M(d) ←ExtractModifiableIdentifiers (d,L);
2  construct a matrix M with |M(d)| rows and two columns;
3  foreach i ∈ M(d) do
4      set the i-th row of M to (i, 0...0 (dependency pattern p) );
5  foreach wis-constraint ψ_j do
6      foreach each i in I(ψ_j, d) do                          /* compute ψ_j */
7          set p[j] to 1 in the i-th row (i, p) of M;
8  sort M according to p and hash(i ∘ K) ;
   // identifiers with the same dependencies look randomly shuffled
9  set cursor to the beginning of M ;
10 repeat
11     (i^1, p^1) ← NextRow (M);
12     (i^2, p^2) ← NextRow (M);
13     if (p^1 = p^2) then                                     /* same dependency */
14         add (i^1, i^2) into pairs;
15 until (end of M) ;
16 return pairs;
```

Algorithm 5: `PairDetect`(suspect document d,$markList$,message length l,key $\mathcal{K}$,ξ)

```
1  for k ∈ {1, ..., l} do
2      vote[k][0]← 0;
3      vote[k][1]← 0;
4  for (i^1, i^2) in markList do
5      j ← S_2(i^1 ∘ K) mod ξ;                                 /* bit index */
6      k ← S_3(i^1 ∘ K) mod l;                                 /* letter index */
7      mask ← S_4(i^1 ∘ K) mod 2;                              /* mask bit */
8      readMark ← v[i^1][j] ⊕ mask;                            /* read the mark */
9      vote[k][readMark]← vote[k][readMark] + 1;
   // voting
10 return ThresholdMajority (vote);
```

Robustness An attacker that performs random alterations is more likely to hit a bit embedded in a pair than a bit embedded at a single position. Therefore a watermark bit embedded in a pair is less robust than a bit embedded at a single position. This slight loss of robustness is traded for computational speed-up and collusion security, as explained in the next section. This phenomenon has to be put into balance with the large amount of pairs discovered by the pairing algorithm. This allows for a large repetition of the bit encoding, which enforces robustness. Experimental evaluations (Section 4.7) assess this property. If an attacker knows which usability constraints are preserved in his watermarked dataset, a more sophisticated attack may be envisioned. By running `ComputePairs` (which is assumed as being public), an attacker might find the dependencies of identified values. But even if he knows that paired tuples share the same dependencies, the exact pairing can not be guessed, because pairs are chosen according to a secret order, known only by the data owner.

Complexity The following table sums up the number of query computations needed to find T distinct watermarks. Parameter n_l denotes the number of linearizable constraints and n_g the number of non-linear constraints. Remember that each query must be computed on a likely huge number of tuples.

Method	#(query computation)	#(ILP solving)
Greedy	$T.(n_g + n_l)$	0
Linear	$n_l + T.n_g$	T
Pairing	$\mathbf{n_l + T.n_g}$	**0**

4.5 Collusion-secure fingerprinting

Suppose that three fingerprints $m_1 = 010, m_2 = 100, m_3 = 110$ are used for the three users u_1, u_2 and u_3. If u_1 and u_2 compare their respective databases, they will observe that they differ on the positions where the first and second bits of m_1 and m_2 have been embedded. Indeed, these bits are different in fingerprints m_1 and m_2. Therefore u_1 and u_2 can guess that the embedding process modified the original database at these precise locations. On the other hand, the positions where the third bit of the fingerprint messages was embedded are not revealed, because this bit is the same in m_1, m_2 and m_3. Suppose now that u_1 and u_2 combine their databases to obtain a third database in which the values located at the modified positions are taken from both databases. Since m_3 has the first bit of m_2 and the second bit of m_1, it can happen that u_1 and u_2 actually build a database that carries the fingerprint of u_3. At detection time, u_3 will be considered as suspect. This kind of coalition of users that collude for framing another one, which is innocent, must clearly be avoided. To achieve this, frameproof collusion-secure codes [17, 95, 110] have been designed. Basically, they must provide the following two features: (i) no coalition of at most c users can frame another innocent user and (ii) there exists a tracing algorithm A that, given a suspect fingerprint, outputs at least one of the members that participated to the framing coalition. A codebook of collusion-secure fingerprints is characterized by the length of its words which depends on the size c of the maximum coalition for which the code is frameproof, the maximum probability ε of error of A and the maximum number M of fingerprints. For instance, this length is $O(c^4 \log(1/\varepsilon) \log(M/\varepsilon))$ for the Boneh and Shaw's scheme [17] ($O(c^4 \log c \log(M/\varepsilon))$ with the improvements of Schaathun [95]) and $100c^2 \log(\lceil 1/\varepsilon \rceil)$ for Tardos' scheme [110]. For instance, if we want to be frameproof against any coalition of at most $c = 3$ purchasers with a probability of error of $\varepsilon = 10^{-4}$, the length of the fingerprints is 3 600. To be effective, there must be some fixed set of positions within the database in which all the codewords can be embedded without affecting the usability. A recurrent problem of collusion-secure codes is that their length tends to increase quite quickly, making them not suitable for all practical fingerprinted applications.

Collusion-secure fingerprinting techniques have already been applied to databases [73] but without taking into account the usability constraints. Query-preserving watermarking is highly challenging since usability constraints drastically reduce the embedding bandwidth. The greedy approach [103] has some limitations that prevent the use of efficient collusion-secure fingerprints. For example, it may embed the codeword for user u_1 in positions, say, i_5 to i_7 while embedding the codewords for u_2 and u_3 in positions i_8 to i_{10}, *i.e.* in completely different positions than for u_1. Hence databases of u_1 and u_2 differ on *all* watermarked positions,

and all the bits of u_1 are exposed. Consequently, the fixed positions requirement is not satisfied by the greedy method. Moreover, this algorithm might actually embed a significantly small number of fingerprints, since it discards the modifications performed by the watermarking algorithm when they do not satisfy the usability constraints. The use of the pairing heuristic enables our algorithm to discover a large mark embedding area, making the collusion-secure fingerprinting possible for large databases whereas the greedy approach does not allow it. Furthermore, the modified positions do not change when different messages need to be inserted. This fits the fixed set of positions required by the collusion secure codebooks that are used. In the implementation of WATERMILL [27], we chose to use the Tardos scheme [110] for the following two reasons: (*i*) codewords have a relatively small length compared to other schemes in our setting (large datasets and a small number of purchasers), (*ii*) the number of users M has not to be a-priori fixed and finally, (*iii*) its implementation is simple and efficient. The codewords of a Tardos scheme are randomly generated according to M independent identically distributed Bernoulli variables whose parameters are functions of M and k.

4.6 Analysis

In this section, we provide an analysis of false hits and false misses occurrence probabilities. Both probabilities highly depend on the detection threshold α. If $\alpha \to 1/2$, the detector is very selective and will lead to a small rate of false hits but at the price of a loss of robustness (many false misses). On the contrary, if $\alpha \to 0$, there will be many false hits but the detection is going to be very robust (very few false misses).

4.6.1 False Hits

A *false hit* is the wrong detection of a watermark in a non watermarked database. In the case of 1-bit fingerprints, we note $v_0 = vote[1][0]$, $v_1 = vote[1][1]$, and $m = v_0 + v_1$. A false hit occurs as soon as $v_0/m > 1/2 + \alpha$ (fingerprint 0 is detected) or $v_1/m > 1/2 + \alpha$ (fingerprint 1 is detected). When $v_0/m, v_1/m \leq 1/2 + \alpha$, no defined fingerprint is detected. For a non watermarked dataset, the value of each $readMark$ (see LinearDetect and PairDetect algorithms) can be modeled as the outcome of a binomial law of parameter 1/2. We define the random variable S as the sum of the m random independent variables, thus modelling the sum of m $readMarks$. Clearly, $E[S] = m/2$ and the probability of false hits is the probability $Pr\left(|S - E[S]| \geq m\alpha\right)$. Using this model, we obtain the following theorem:

Theorem 48 *For all $\delta > 0$, if $\alpha \geq \alpha^+(m) = \sqrt{\frac{-\log \delta}{2m}}$, then false hits occurrence probability (for one bit fingerprints) is less than δ.*

PROOF. It has been shown by Hoeffding [50] that, for m independent random variables $X_1, .., X_m$ that are bounded ($a_i \leq X_i \leq b_i$), their sum S satisfies the following inequality for any $t > 0$: $Pr\left(|S - E[S]| \geq mt\right) \leq \exp\left(-\frac{2m^2t^2}{\sum_{1,m}(a_i - b_i)^2}\right)$. Applied here with $t = \alpha$, $a_i = 0$ and $b_i = 1$ this inequality gives: $Pr\left(\left|S - \frac{m}{2}\right| \geq \alpha m\right) \leq \exp\left(-\frac{2m^2\alpha^2}{\sum_{1,m} 1^2}\right) = \exp\left(-2m\alpha^2\right)$, whence $Pr\left(\left|\frac{S}{m} - \frac{1}{2}\right| > \alpha\right) \leq Pr\left(\left|\frac{S}{m} - \frac{1}{2}\right| \geq \alpha\right) \leq \exp\left(-2m\alpha^2\right)$. Hence, if $\alpha \leq \alpha_0 = \sqrt{\frac{-\log \delta}{2m}}$, $\exp\left(-2m\alpha^2\right) \leq \exp\left(-2m\alpha_0^2\right) = \delta$. □

4.6.2 False Misses

A *false miss* occurs when the detected message on a modified dataset (*e.g.* an attacked one) contains undefined bits or when it is different from the embedded one. Here, we perform an analysis in the 1-bit fingerprint case. We model attacks using the model of p-attacks. A p-attack is an attack for which each $readMark$ has the probability p to be inverted. Common attacks like random bit-flipping and translation attacks are captured by this model (for different values of p though). Suppose that the embedded fingerprint is 0. Then, a false miss occurs on the attacked dataset if $v_0/m \leq 1/2 + \alpha$. If $v_1/m > 1/2 + \alpha$, the invalid message 1 is detected. If $v_1/m \leq 1/2 + \alpha$, an 'undef' message is detected.

Theorem 49 *For all* $0 < p < \frac{1}{2}$*, if* $\alpha \leq \alpha^-(m) = \frac{1}{2} - p - \sqrt{\frac{-\log \delta}{2m}}$ *and* $m \geq -\frac{8 \log \delta}{(1-2p)^2}$*, then the false miss occurrence probability (for* 1*-bit fingerprints) over* p*-attacked database is less than* δ*.*

PROOF. Let $0 < p < \frac{1}{2}$, $\alpha = \frac{1}{2}(\frac{1}{2} - p)$ and $t = (\frac{1}{2} - p) - \alpha$. Hence, $t = \frac{1}{2}(\frac{1}{2} - p) = \alpha > 0$. In the context of a p-attack, the probability that each $readMark$ is preserved is $1 - p$. Therefore the intended value for $\frac{v_0}{m}$ is $1 - p$ and we can quantify the probability of diverting from this value using the Hoeffding inequality:

$$Pr\left(\left|\frac{v_0}{m} - (1-p)\right| \geq t\right) \leq \exp\left(-2mt^2\right).$$

Let $x \in \left[0, \frac{1}{2} + \alpha\right]$. Then $x \leq \frac{1}{2} + \alpha$. Since $\frac{1}{2} + \alpha = 1 - p - t$, $x \leq 1 - p - t$ and $x - (1-p) \leq -t < 0$. Hence, $|x - (1-p)| = (1-p) - x \geq t$. In other words $x \leq \frac{1}{2} + \alpha \Rightarrow |x - (1-p)| \geq t$. From this implication we obtain that:

$$Pr\left(\frac{v_0}{m} - \frac{1}{2} \leq \alpha\right) \leq Pr\left(\left|\frac{v_0}{m} - (1-p)\right| \geq t\right) \leq \exp(-2mt^2).$$

Suppose now that $m \geq -\frac{8 \log \delta}{(1-2p)^2}$. Then,

$$\exp\left(-2mt^2\right) \leq \exp\left(2\frac{8 \log \delta}{(1-2p)^2}t^2\right) \leq \delta$$

and

$$Pr\left(\left|\frac{v_0}{m} - 1/2\right| \leq \alpha\right) \leq \delta.$$

□

Example Suppose that $\xi = 3$. If an attacker randomly selects one of the ξ least significant bits and inverts it, the probability for each watermarked bit to be inverted is $1/3$. If we want to have both false hits and false misses occurrence probabilities below δ, we need to have $\alpha^+(TC) \leq \alpha \leq \alpha^-(TC)$, which can be achieved using $\alpha = \frac{1-2p}{4}$ as soon as $\alpha^+(m) \leq \alpha^-(m)$, *i.e.* when $m \geq -\frac{16 \log \delta}{(1-2p)^2}$. For $\delta = 10^{-3}$, we obtain $m \geq 994.71$. If the database contains $10,000$ tuples, setting $\gamma = 10$ $\left(\simeq \frac{10000}{994.71}\right)$ for watermarking achieves the target robustness.

4.6.3 Extension to Arbitrary Length Fingerprints

Suppose in the following that the embedded fingerprints have the length l. We note $m_i = vote[i][0] + vote|i][1]$.

Corollary 50 *For all* $\delta > 0$*, if* $m = min(m_i)$ *and* $\alpha \geq \alpha^+(m) = \sqrt{\frac{-\log \delta}{2lm}}$*, the probability that a fully defined message is detected in a third party dataset is less than* δ*.*

PROOF. For $i = 1 \ldots n$, the probability that the bit i of the detected message is defined is at most $\exp(-2m_i\alpha^2) \leq \exp(-2m\alpha^2) \leq \delta^{1/l}$. Then, for a message of l bits, the probability that all l bits are defined is at most δ. □

Notice that this corollary deals only with detecting a fully defined message. Not every message may be valid if the owner has only distributed N copies. In this case, the probability that a *valid* fingerprint is recovered is at most $\frac{N}{2^l}\delta$. An interesting case is the one of the p-attacks on datasets in which fingerprints of length $l > 1$ were embedded. In the 1-bit case, p-attacks when $p < 1/2$ were not discussed. Indeed, they were likely to invert the fingerprint and thus avoid detection. When fingerprints contain several bits, if $p < 1/2$, then all the bits of the fingerprint are very likely to be *simultaneously* inverted. In that case, the recovered fingerprint may seem as suspect as the original one (think of it as a negative image of a black and white picture). Hence, attacks are going to be effective only if $p = 1/2$ which is *possible* (*e.g.* by inverting all bits of the dataset) but at the same time *destructive* since all the values are altered.

4.7 Experimental Results

Context Our methods are implemented on our open platform WATERMILL [27]. The experiments were performed on a workstation running Debian GNU/Linux (AMD64 port). Hardware includes an EMT64 P4@3.2Ghz HT Intel processor, 1Gb of RAM and a 80Gb 7200RPM 8MB cache SATA hard drive. We use Mysql version 5.0.18 (Debian packaged), Sun's Java J2RE 1.4.2 Standard Edition (Hotspot 64-Bit Server VM), and JDBC is supported through mysql-connector-java version 3.1.6. No special hardware nor software (except for the WATERMILL prototype) tuning was performed. Mysql databases use MyISAM, *e.g.* a transaction-less physical storage engine. Swap space is 2Gb.

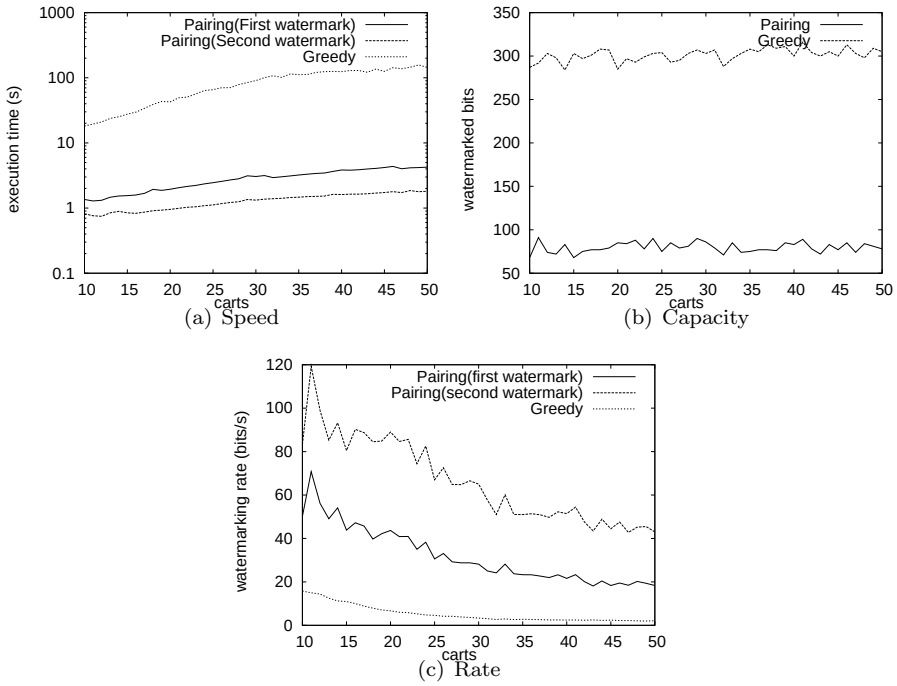
Benchmark datasets Experiments were performed on two different datasets: a synthetic dataset and the Forest CoverType database from the UCI KDD archive [5]. The *synthetic relational database* corresponds to a sales database, with n *products*, each product having an associated *cost*. A number of p *shopping carts* are filled with random subsets of k products. We denote such an instance by $B(n, p, k)$. For different values of n, p and k, we modified the *cost* attribute, with the following usability constraints: the distortion on the cost of each product must not exceed 1 Euro, and the distortion on the total cost for *each shopping cart* must not exceed 1 Euro.

Observe that for an increasing number of carts, these constraints are very restrictive and hard to respect simultaneously, even on a small dataset. The second set of experiments was performed on the *Forest CoverType database* [5]. This database gathers information on forest parcels, for a total of 581,012 tuples. We have restricted our attention to the `elevation` and the `aspect` attributes. We created virtual primary keys for the dataset that do not exist in the original dataset. We watermarked the *aspect* attribute, with a local distortion of 1. We split the *elevation* values into 50 random overlapping intervals. The 50 corresponding usability constraints impose that the *mean (i.e sum) of aspects* of data with *elevation* in the same interval must *not* be altered by more than 1 unit (meaning maximal global distortion of 1).

4.7.1 Speed and Capacity

4.7.2 Synthetic Dataset

For the instance $B(5000, 1000, 3)$, checking the global constraints takes on average 145s. If $\gamma = 10$ (one tuple out of ten is altered), the global constraints are going to be checked about 500 times for the greedy method, requiring more than 20 hours. Remark that $B(5000, 1000, 3)$ is not a large database. Consequently *for huge datasets, using the greedy method is impossible since it requires to perform costly computations each time a watermark bit is to be embedded*. For our experiments, we used only small datasets to be able to make a comparison between the two methods. The *greedy* method refers to the method of Agrawal, Haas and Kiernan [7] with a check of the constraints every time a bit is modified. If the constraints are respected, the modification is accepted otherwise it is discarded. To compare speed and capacity, we performed a series of experiments, using the instance $B(1000, p, 3)$ *i.e.* 3 products per cart. For a number of carts p ranging from 10 to 50, we compared *watermarking times*, the *number of watermarked bits* and *watermarking rates* (number of valid watermarked bits per second) for both *greedy* and *pairing* algorithms. For each experiment, two values were recorded: the time to obtain a watermarked database and the number of altered bits. A higher number of bits is better because it allows for a larger embedding bandwidth. For the pairing method, two time values were recorded: the highest represents the precomputation of the pool whereas the lowest is the time to obtain a watermarked database once the pool has been precomputed. Results are presented in Fig. 4.3, watermarking speed using a logarithmic scale on Fig. 4.3(a), watermarking capacity on Fig. 4.3(b) and the watermarking rate on Fig. 4.3(c). Clearly, the pairing algorithm outperforms the greedy one from the speed point of view. It can be argued than significantly more bits can be hidden using the greedy method. Indeed, the capacity is 3 to 4 times higher for the greedy method. There are several reasons for this. First, bits are altered by pairs (a factor 2). Second, not all pairs are altered, only the one having different binary values at the watermarked positions (statistically, another factor 2). Third, the pairing algorithm does



(a) Speed (b) Capacity (c) Rate

Figure 4.3: Comparison of the greedy and pairing algorithms on a synthetic database

Table 4.1: Comparison of greedy and pairing methods on a real database

	Greedy method	**Our method**
γ	8	3
Hidden bits	39813	48052
Watermark density	6.9%	8.27%
Mean square error	0.0059	0.0825
Precomputation	negligible	8min15s
Obtaining the first mark	62h30min	8min15s + 5min53s.
Obtaining a new mark	**62h30min**	**5min53s.**

not find all possible alterations but only a subset of them, as the greedy method do. Nevertheless, the watermarking rate is about 10 times better for the pairing method.

4.7.3 Forestcover Dataset

With the Forest CoverType dataset, experiments show that more than 70000 bits can be embedded using the pairing algorithm (with $\gamma = 8$). To reach the same number of watermarked bits, the greedy method would have to check the constraints at least 70000 times, which is expected to be very slow. Table 4.1 presents the results. The greedy method requires more than 2 days of computation whereas our method needs only a few minutes. Situation becomes worse when another fingerprinted instance with the same constraints has to be obtained. Suppose that we want to distribute fingerprinted copies to 4 different purchasers. Here, the fingerprints can be coded as binary strings of length 2. To each one of the n valid bit embedding positions is randomly mapped one of the two bits of the fingerprint. Then, it can be shown that, on average, the whole fingerprint is embedded $m \simeq \frac{n}{2}$ times (*i.e.* both bits 1 and 2 of the fingerprint are embedded more than m times). In our example, for 4 purchasers, the fingerprint is redundantly embedded 24026 times using the

pairing method and 19906 times using the greedy one. Going back to our introductory example, if the value time-window of the Forest CoverType dataset is one week, it is not possible to distribute it to 4 customers using the greedy method. Indeed, each fingerprinted instance requires more than 2 days of computation, *i.e.* more than 8 days for 4 customers.

4.7.4 Robustness

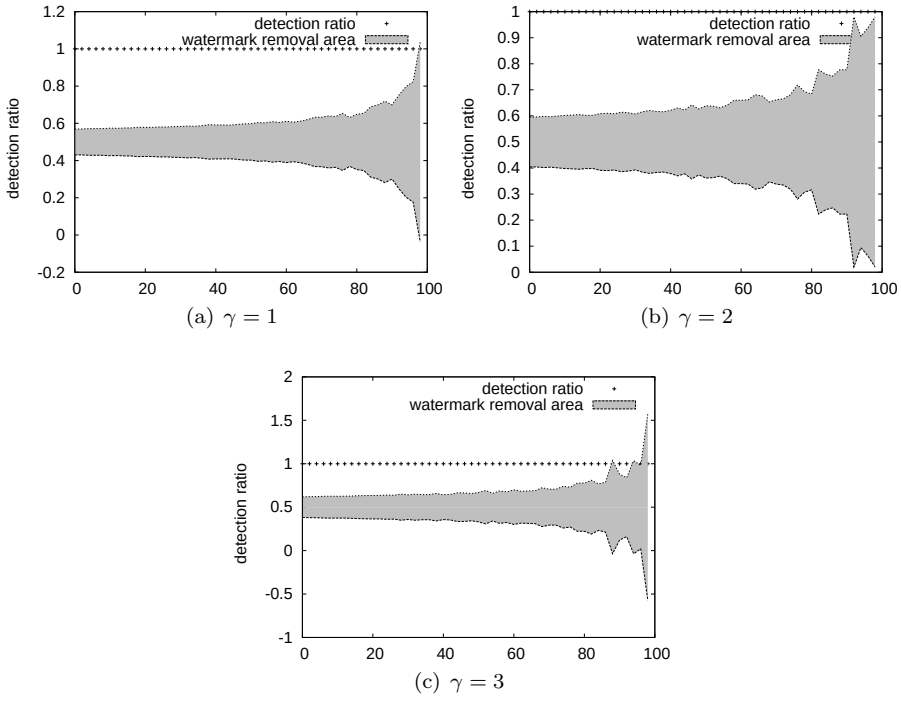
An attack is considered as effective if it successfully cheats the detector while introducing a distortion on the data comparable to the distortion introduced by the watermarking process. Here, we measure the distortion by the mean of squared errors mse (we do no use the mean since it is not modified by the pairing algorithm). If d and d' are two databases such that $\mathcal{I}(d) = \mathcal{I}(d')$, $mse(d, d')$ is defined as follows ($N = |\mathcal{I}(d)|$):

$$mse(d, d') = \frac{1}{N}\sum_{i=1}^{N}(\vec{v}(d)[i] - \vec{v}(d')[i])^2.$$

For a database $\tilde{d}$ obtained by watermarking d, $mse(d, \tilde{d}) = \|\vec{w}\|^2/N$.

4.7.5 Subset Attacks

The subset attack consists of discarding from the relation every tuple with a probability q. Figure 4.4 shows the detection ratio of watermarked bits against the value of q for an instance $B(1000, 100, 5)$. For each experiment the threshold value α was chosen so that the false hit occurrence probability remains below 0.1%. Points within a radius α of 1/2 have been colored in gray. They represent the watermark removal area. Observe that the detection ratio remains 1 for all attacks. Note also that, in order to keep a false hit occurrence probability under 0.1%, α increases with q. The attacks always fail unless $\gamma = 3$ and $q > 80\%$, *i.e.* when the attack discards almost the entire dataset.



(a) $\gamma = 1$ (b) $\gamma = 2$

(c) $\gamma = 3$

Figure 4.4: Watermark detection after subset attacks

4.7.6 Data Alteration Attacks

Another kind of attack consists of modifying the values within the dataset. Remark that such modifications are likely to break usability constraints. An ε-attack is an attack where a bounded random distortion is added to a randomly selected set of tuples. The maximum distortion is called the amplitude and is noted ε. To each tuple (watermarked or not) is added a random distortion d ($0 \leq d < \varepsilon$) with a probability i. We followed the protocol, with $\xi = 1$:

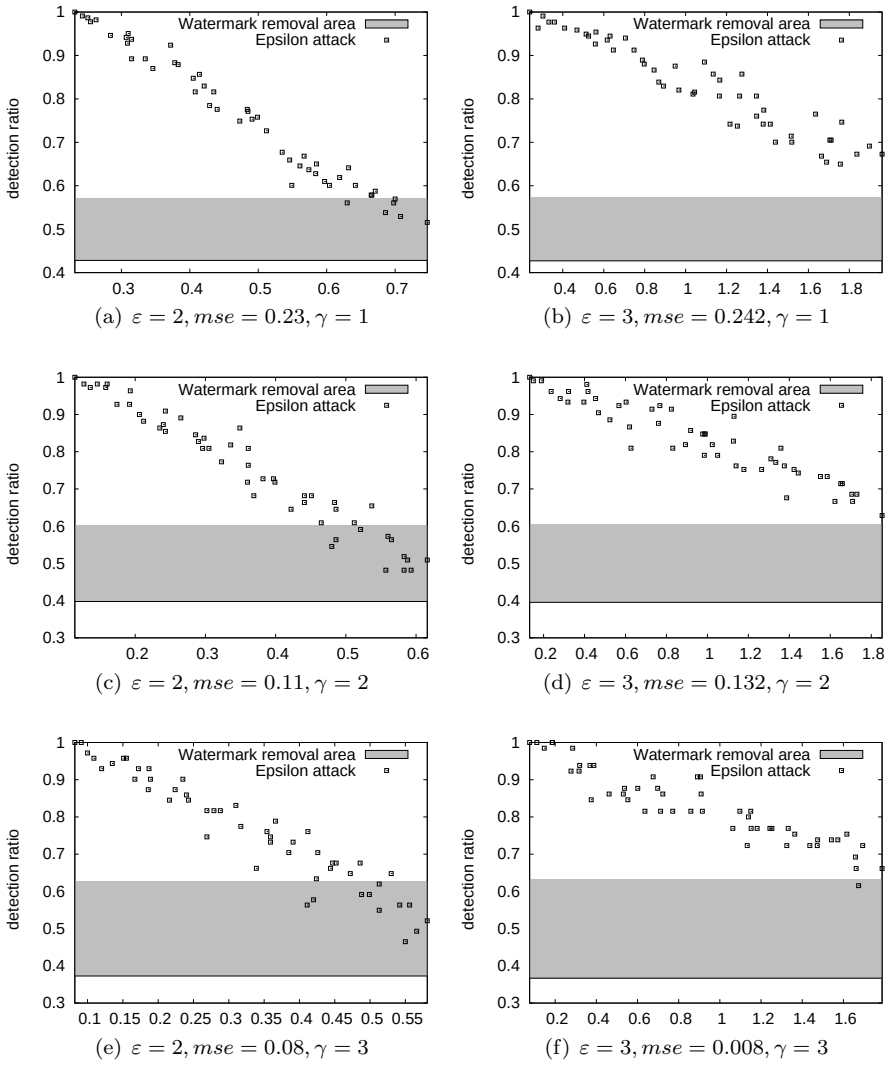
1. Create the instance $I = B(1000, 100, 5)$.
2. Precompute the constraints on I.
3. For each $(\varepsilon, \gamma) \in \{2, 3\} \times \{1, 2, 3\}$:
   - get a watermarked instance I_γ with insertion rate γ;
   - for each value of ε and each value of $i \in 0 \ldots 100$, get an attacked instance $I_{\gamma,\varepsilon}$;
   - compute the bit detection ratio x and the mean squared mse error after the attack;
   - plot (mse, x).

Results of experiments (see Fig. 4.5) are represented with the mse of the attack displayed horizontally, and the detection ratio vertically. The watermark removal area is displayed as a gray rectangle (chosen here so that the false positive occurrence probability is at most 0.1%). Graphs are labelled by the mse of the watermarking process. Successful (resp. unsuccessful) attempts to remove the watermark are pictured by dots inside (resp. outside) the gray rectangle. The majority of successful attacks is observed for $\varepsilon = 2$ (only one attack is successful for $\varepsilon = 3$). This can be explained by the fact that the lower ε, the higher is the probability for an alteration to 'hit' the watermarked bit. Second, if the attacker wants to erase the mark, he has to alter the quality of the data significantly more than the watermarking process. For instance, when $\gamma = 2$ and $\varepsilon = 2$ (Fig. 4.5(c)), first successful attacks are observed for a mean squared error of 0.5 whereas the watermarking process introduced an error of 0.11. Once more, the price an attacker has to pay for a perfect watermark removal is significantly higher than the one paid for watermarking.

Collusion-secure fingerprinting On the Forest Covertype benchmark, the pairing heuristics identifies 48 052 embedding positions in roughly 14mins. By setting $\gamma = 4\ 200$, the execution of the greedy methods takes roughly the same amount of time but locates only 93 embedding positions. Assuming that each bit of the fingerprint is embedded 5 times, we can embed fingerprints that have length $48\ 052/5 \simeq 10\ 000$. Using the Tardos codebook [110], the maximum size c of a coalition of users against which the scheme is frameproof can be computed as $c = (l/(100 \log(\lceil 1/\varepsilon \rceil))^{1/2}$. For $\varepsilon = 10^{-4}$, we obtain $c \simeq 3.3$. Fingerprints are then frameproof against a coalition of 3 users.

4.8 Related Work

Several recent works consider relational databases watermarking [7, 39, 71, 73, 103]. Agrawal, Haas and Kiernan's method [7] hides information in the least significant bits (LSB) of numerical attributes. The database owner can control the alteration on attributes by setting the number of LSB that can be modified. Although a small overall distortion on the mean of the watermarked attributes is observed, more general usability constraints are not considered, like the ones preserving the result of important SQL queries. Their technique was extended [73] to collusion-resilient fingerprinting by using collusion-secure codebooks [17]. But again, usability constraints are not handled. Sion, Atallah and Prabhakar [103] introduced the greedy method for watermarking with usability constraints. They handle potentially any kind of constraints by calling external checking programs (*usability plugins*). This very general method is not optimized as explained in the introduction, since the syntactical form of usability constraints is not explored (but another kind of optimization is considered in [102]). It can also be applied to fingerprinting, but the computational



(a) $\varepsilon = 2, mse = 0.23, \gamma = 1$ (b) $\varepsilon = 3, mse = 0.242, \gamma = 1$

(c) $\varepsilon = 2, mse = 0.11, \gamma = 2$ (d) $\varepsilon = 3, mse = 0.132, \gamma = 2$

(e) $\varepsilon = 2, mse = 0.08, \gamma = 3$ (f) $\varepsilon = 3, mse = 0.008, \gamma = 3$

Figure 4.5: Watermark detection after subset ε-attacks

effort is tremendous. Combining their method with collusion-secure codebooks is also possible but with some limitations: there is absolutely no guarantee that the same watermark positions will be found for all watermark messages in the codebook since their method is greedy. Besides, it is noteworthy that their embedding method, that differs from the LSB embedding introduced by Agrawal, Haas and Kiernan, can be adapted to our approach with a small effort. Weight-independent *sum* constraints were introduced from the theoretical point of view by one of the authors of the present work in [39]. This specific query pattern was studied in order to obtain a lower bound on the number of distinct acceptable watermarks that one is likely to discover. The algorithmic counterpart of this previous paper is less suited for practical applications. The present work considers algorithms that behave correctly with large datasets. Moreover, the algorithms of this previous work were not *blind*, while our new algorithms are. Finally, collusion attacks and other constraints described in the current chapter were not considered by [39].

4.9 Conclusion

It this section, we present an optimization technique for the discovery of good watermarks in a dataset that respects several usability constraint patterns. We have also considered the problem of collusion-secure fingerprinting under these constraints. Natural extensions of this work are the following. First, the number of our constraint patterns could certainly be increased. Second, we would like to address databases fingerprinting where purchasers do not share the same usability constraints. Finally, we would like to devise tools for proving ownership on datasets after a specific rewriting.

Related publications

- **Julien Lafaye, David Gross-Amblard, Camélia Constantin and Meryem Guerrouani.** WATERMILL**: an optimized fingerprinting system for highly constrained data.** ***IEEE Transactions on Knowledge and Data Engineering (TKDE)*****, 20(4): 532-546, April 2008.**
- Camélia Constantin, David Gross-Amblard and Meryem Guerrouani. Watermill: an Optimized Fingerprinting Tool for Highly Constrained Data. In *ACM Workshop on Multimedia and Security (MMSec)*, New-York, USA, August 1-2 2005, pp. 143-155, 2005.
- (soft) Camélia Constantin, David Gross-Amblard, Meryem Guerrouani et Julien Lafaye. WATERMILL: an optimized watermarking/fingerprinting tool for databases.
  `http://watermill.sf.net`

5
Typed XML streams

XML streams are online, continuous, distributed, *high throughput* sources of information. For streams carrying a significant intellectual and/or industrial value, proving ownership over pirated copies is a requirement for data producers. While watermarking methods already exist for numerical streams, they do not meet the specific requirements of XML streams. In this chapter, we introduce the ℓ-`détour` algorithm, which allows for watermarking XML streams so that (i) the watermark embedding and detection processes are done online and use only a constant memory, (ii) the stream distortion is controlled, (iii) the type of the stream is preserved and finally (iv) the detection procedure does not require the original stream.We also evaluate, analytically and experimentally, the robustness of ℓ-`détour` against attempts to remove the watermark.

5.1 Introduction

Streams. Data streams are *high throughput* sequences of *tokens*, potentially *infinite*. They are used in a growing number of applications (see e.g. [12]) and their specificities make them a challenging application [76]. Since XML has become the standard for specifying exchange formats between applications, the focus in this chapter is on XML streams. XML streams are commonly available online and processed by distant peers. Data *producers*, e.g. news providers, generate the tokens of the stream which is later on processed by *consumers*. Consumers do not accept arbitrary streams, but place restrictions on their *input types*. For XML based systems, types are usually specified through a Document Type Definition (DTD) or an XML Schema. High throughput requirement puts severe constraints on consumers: they must be able to process each token of the stream quickly and cannot buffer an arbitrary number of tokens (bounded memory). For any arbitrary DTD, typechecking XML streams can not be done while respecting these constraints. Hence, we focus on *acyclic* DTDs , where no element is a sub-element of itself (for example, RSS is an acyclic DTD). Under this hypothesis, typechecking can be done using deterministic finite automata (DFA) and types can be specified using regular expressions [101].

Example 51 *The* XML *news feed of Fig. 5.1 may be regarded as a stream on an alphabet of closing and ending tags (<* `news` *>, < /`date` >..), letters (S,o,d,e,1,...) and predefined sequences of letters (Cinema, Politics, ...). It can be typechecked using the regular language* `<news><priority>[123]</priority><title>(.*)</title>..<date>D</date>...</news>`*, where the expression* `D=(19|20)[0-9][0-9]-(0[1-9]|1[0-2])-(3[0-1]|0[1-9]|[1-2][0-9])` *captures valid dates (for the sake of simplicity we do not try to check dates like* `2005-02-31`*). Observe that the* DTD *standard does not allow the definition of a precise date format, since the contents of elements are mostly of type PCDATA (i.e. almost any sequence of letters). A more sophisticated model like* XML Schema *allows for such precise definitions. Our model applies to both formalisms.*

Watermarking. High-quality streams carry a great intellectual and/or industrial value. Malicious users may be tempted to make quick profit by stealing and redistributing streams illegally. Therefore, data producers are interested in having a way to prove their ownership over these illicit copies. *Watermarking* is known to bring a solution to that issue by hiding copyright marks within documents, in an imperceptible

```
...</news><news>
  <priority>1</priority>
  <title>Soderbergh won the Golden Palm</title>
  <url>http://www.imdb.com/title/tt0098724/</url>
  <date>1989-05-23</date>
  <text>Soderbergh's movie, Sex, lies and videotapes, won the ...</text>
  <category>Cinema</category>
</news><news>...
```

Figure 5.1: An XML stream snapshot

and robust manner. It consists of a *voluntary alteration* of the content of the document. This alteration is parameterized by a key, kept secret by the owner. Accordingly, the secret key is needed to detect the mark and thus, to prove ownership. The robustness of the method relies on the key in the sense that removing the mark without its knowledge is very difficult. A first challenge of streams watermarking is to control and minimize the alteration of the stream, i.e. to *preserve* its *quality*. We measure the alteration by means of a relative edit-distance and propose a watermarking algorithm that introduces a bounded distortion according to this measure. A second challenge is to *preserve the type of the stream* so that it remains usable by its intended consumers. Existing XML watermarking schemes embed watermarks by modifications of the content of text nodes. We believe that other embedding areas may be used, e.g. within the tree-like structure itself. Obviously, altering the structure can not be done naïvely. For instance, in some text watermarking schemes, bits are embedded by switching words of the document with their synonyms. This can not be directly applied to our context: if the name of an opening tag is switched, the corresponding closing tag has to be switched to ensure well-formedness. Even if tag names are switched consistently, the resulting document may become invalid with respect to its original type. In that case, watermarked documents are *unusable* by their target consumers. Remark also that a good watermarking method must be *robust*, i.e. still detects marks within streams altered (random noise, statistical analysis, ..) by an attacker (up to a reasonable limit).

Our Contribution. In this work, we introduce the ℓ-`détour` algorithm, a robust ε-preserving watermarking scheme for XML streams, valid with respect to acyclic DTDs. The idea of ℓ-`détour` is the following. We identify two relevant parts of the stream, based on its semantics. The first *unalterable* part can not be altered by any attack without destroying the semantics of the stream. The second *alterable* part is still useful for the application, but can be altered within reasonable limits. For the automaton of Figure 5.1, the *unalterable* part will be e.g. the path name in the `url` element (but not the host name, since it can easily be replaced by an IP number). The alterable part will be e.g. the two digits of the day in the `date` element. Alterable parts can capture purely textual information as well as structuring one. A finite portion of the *unalterable* part, combined with a secret key known only by the data owner, is used to form a *synchronization key*. A non-invertible (cryptographic) pseudo-random number generator, seeded with this synchronization key, determines how the *alterable* part of the stream is modified to embed the watermark. This process, repeated along the stream, introduces *local dependencies* between parts of the data stream. These dependencies, invisible to anybody who does not possess the key used for watermarking, are checked at detection time by the owner. Only the private key and the suspect stream are needed. It can be viewed as an extension of Agrawal and Kiernan's method [7] which considered relational databases watermarking (primary keys played the role of our synchronization keys). In order to respect the type constraint, we simulate the DFA that typechecks the stream. Each time the insertion of a dependency is required, we change a sequence of tokens of the stream so the walk on the automaton follows a *detour*, leading to the *same* state. If the altered sequence lead to state q, the chosen detour still leads to q. The length ℓ of the detours and the frequency of the alteration control the quality of the stream. The DFA is also used to define the alterable and unalterable parts of the stream.

Contribution. In Section 5.2, we present our main contribution: the ℓ-`détour` algorithm, which allows for watermarking XML streams so that (i) the watermark embedding and detection processes are done online

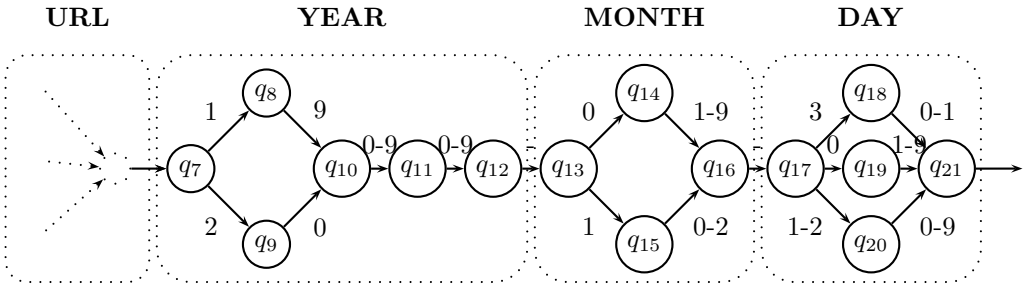


Figure 5.2: A partial specification of the stream type for news items (`date` element)

and use only a constant memory, (ii) the stream distortion is controlled, (iii) the type of the stream is preserved and finally (iv) the detection procedure does not require the original stream. In Section 5.3, we discuss on the robustness of ℓ-`détour` against attempts to remove the watermark and show that attackers have to alter more the streams than the watermarking process did to remove the mark. Comparison with related work is presented in Section 5.4. Section 5.5 concludes.

5.2 The ℓ-`détour` Algorithm

5.2.1 Preliminaries

In this chapter, we use ω-rational languages on words, i.e. a simple, yet expressive, extension of regular languages suited to infinite words.

- **Streams**: *Let Σ be a finite alphabet. Letters from Σ are called tokens. A Σ-stream σ is an infinite sequence of tokens from Σ.*
- **Stream Automaton**: *A stream automaton is a deterministic finite state automaton such that all states are accepting, except one which has no outcoming edge to an accepting state. This state is called the blocking state.*
- **Stream Acceptance**: *Let G be a stream automaton. A stream σ is accepted by G if the walk on G due to σ never enters the blocking state.*
- **Stream Types**: *A set of streams $\mathcal{L}$ is a stream type if there exists a stream automaton G such that $\mathcal{L}$ is the set of all streams accepted by G.*

Example 52 *Figure 5.2 shows a partial specification of a stream automaton for the input type of a news items consumer. It checks that the syntax of the date is correct. The part checking that the stream is well-formed and conforms to the complete* DTD *is not depicted here. All unspecified transitions lead to the blocking state.*

As a means to measure the distortion introduced by watermarking algorithms, we introduce the *relative edit-distance*. It is based on the edit-distance for strings [69]. In our context, the edit-distance $d_e(x, y)$ between words x and y is defined as the minimum number of operations (substitution/deletion/insertion of a token) that are needed to transform x into y. For instance, if y has been obtained by substituting one symbol of x, $d_e(x, y) = 1$. The *relative edit-distance* between x and y is defined as the average number of operations per symbol that are needed to transform x into y. We measure the *relative edit-distance* from finite prefixes of streams:

Definition 53 (Distance) *Given σ^N (resp. σ'^M) a finite initial segment of a stream of length N (resp. M), the relative edit distance $d(\sigma^N, \sigma'^M)$ is defined by:*

$$d(\sigma^N, \sigma'^M) = \frac{d_e(\sigma^N, \sigma'^M)}{\sqrt{N}\sqrt{M}}.$$

Example 54 *$d(babba, dabba) = 1/5$. Letter b has been substituted for d (edit-distance 1), and both words have length 5.*

5.2.2 Informal Introduction to ℓ-détour

Suppose that we want to watermark a data stream σ flowing from a producer $\mathcal{P}$ to a consumer $\mathcal{C}$ which input type is specified by a stream automaton G. Since $\mathcal{P}$ produces a usable stream for $\mathcal{C}$, its outputs correspond to non blocking walks on G. Assume that there exist in G two different edges (paths of length 1), labelled by different tokens, and having same start and same end (for example, paths from q_{17} to q_{20} in Fig. 5.2). These edges can be loops on a single node. The idea of our algorithm is to change the value of some tokens of the stream so that the walk on G follows one of these edges rather than the other (for instance, $q_{17} \xrightarrow{1} q_{20}$ instead of $q_{17} \xrightarrow{2} q_{20}$). These tokens are chosen as a function of (1) the secret key K_{p} of the owner and (2) a finite portion, carefully chosen, of the path previously covered. The original walk on the automaton is diverted, and becomes specific to the data owner. This process is repeated along the stream. Notice that following an edge once does not imply that it will always be chosen because the path previously covered varies. Then, a watermarked stream is composed of alternated sequences of *unaltered* segments (*synchronization* segments) and *altered* segments of length 1. The value of an altered segment cryptographically depends on the value of its preceding synchronization segment. This method ensures that the type of the stream is respected. Furthermore, the modified stream is close to the original: each choice between two different paths adds at most 1 to the *edit-distance* between the original and the watermarked stream (and less to the relative edit-distance).

5.2.3 Finding Detours

The previous paragraph gave the idea of the 1-détour algorithm because paths of length 1 were altered in order to embed the watermark. The extension of this algorithm to path of length exactly ℓ is given the name of ℓ-détour. In ℓ-détour, not all paths of length ℓ may be changed but only those called *detours*:

Definition 55 (Detours) *Let G be a stream automaton. The path $p = q_i \to ... \to q_j$ is a detour of length ℓ in G if its length is ℓ and if there is no path p' in G, distinct from p, of length at most ℓ, having the same end points q_i and q_j, and an internal node in common.*

Example 56 *In any stream automaton, all edges are detours of length 1 since they do not contain any internal node. Remark also that as soon as $\ell > 1$, cycles are not allowed in detours of length ℓ. On the automaton of Fig. 5.2, there are detours of length 2: $q_7 \xrightarrow{1} q_8 \xrightarrow{9} q_{10}$ and $q_7 \xrightarrow{2} q_9 \xrightarrow{0} q_{10}$. Conversely, paths from q_{13} to q_{16} going through q_{14} are not detours because q_{14} is an internal node common to 9 paths of length 2 between q_{13} and q_{16}. There are 9 paths from q_{14} to q_{16} labeled by 1 to 9.*

The proof of proposition 57 provides a constructive way to compute detours. Due to space reasons, it is not detailed. Remark that space complexity of the method is $O(n^2|\Sigma|\ell)$ whereas it is usually $O(n^2|\Sigma|^\ell)$ to compute paths (and not detours) of length ℓ.

Proposition 57 *Let Σ be the alphabet, n the number of states of the automaton and $\ell \in \mathbb{N},\ \ell > 0$. Detours of length ℓ can be computed in space complexity $O(n^2|\Sigma|\ell)$ and time complexity $O(n^3\ell)$.*

Proof. (sketch) Since detours are paths, i.e. finite sequences of labelled edges, a first naïve strategy is to compute the set of paths of length ℓ and remove paths which are not detours. If $S^k(i,j)$ is the set of paths of length k between states i and j, the formula $S^{k+1}(i,j) = \bigcup_{q \in states(G)} S^k(i,q) \times S^1(q,j)$ permits to define an iterative algorithm to compute $S^k(i,j)$ for any $k > 0$ (if R, S are two sets, $R \times S$ is defined as the set containing the concatenation of every item of R with every item of S). Unfortunately, this leads to an exponential blowup because the number of paths of length ℓ is $n|\Sigma|^\ell$ in the worst case. This blowup can be avoided by getting rid of paths which will not become detours, at each iteration. Indeed, if p, p' are two

detours having the same end points and e is an edge in G, $p.e$ and $p'.e$ are not detours because they share an internal node: $end(p) = end(p')$. This fact remains true for any two paths which have p and p' as prefixes. Similarly, if p is a detour of length k between i and q and e, e' are two edges between q and j, $p.e$ and $p.e'$ are not detours. Hence, we can reduce the number of paths which are detours in the sets computed by the naïve algorithm by modifying the definition of the $\times$ operator: if R and S are not singletons, $R \times S = \emptyset$. This can be checked in constant time. Another condition is necessary to strictly compute sets of detours: if p_1 (resp. p_2) is the only detour of length $k > 1$ between states i and q_1 (resp. q_2) and e_1 (resp. e_2) is the only edge between states q_1 (resp. q_2) and j, $p_1.e_1$ and $p_2.e_2$ are detours of length $k+1$, unless p_1 and p_2 share their first edges. To check this when computing $R \times S$, buffering only the first edge of each path in R is needed. There are at most $|\Sigma|$ such edges.

This leads to a time complexity $O(n^3\ell)$ and a space complexity $O(n^2|\Sigma|\ell)$. At each of the ℓ iterations, there are n^2 sets of detours to compute, each step requiring at most n operations. Space complexity is $O(n^2|\Sigma|\ell)$ because the number of detours is at most $|\Sigma|$ between any two states (two detours can not begin with the same edge). There are n^2 pairs of states and the maximum length of a detour is ℓ. □

5.2.4 Watermark Embedding

The ℓ-`détour` algorithm can be divided into three successive steps. Steps (1) and (2) are performed once for all, while step (3) is used online and requires constant memory.

(1) *Precomputation* of the detours given a target detour length ℓ.

(2) *Annotation of the automaton.* The set of detours is split up into the set of *alterable* ones and the set of *unalterable* ones. Among the set of remaining edges (i.e. edges not part of a detour or part of an unalterable detour), a subset of *synchronization* edges is selected.

(3) *On-the-fly watermarking.* The stream is continuously rewritten by substituting some sequences of ℓ tokens.

STEP 1: Precomputation. For a given input type, a canonical choice for the stream automaton is the minimal deterministic recognizer of the DTD, but any equivalent deterministic recognizer may be used. A strategy is to start with the minimal one and to compute the detours using Prop. 57. If their number is too small or if they do not fit the owner's needs, the automaton can be unfolded into an equivalent one by splitting nodes and duplicating edges, and detours recomputed.

STEP 2: Annotation of the automaton. Not all detours are suitable for watermarking. For instance, on Fig. 5.2, there are two detours of length 2 between states q_7 and q_{10}: $q_7 \xrightarrow{1} q_8 \xrightarrow{9} q_{10}$ and $q_7 \xrightarrow{2} q_9 \xrightarrow{0} q_{10}$. Using these detours for watermark embedding would imply changing the millennium of a news item, resulting in an important loss of semantics. A solution is to divide the previously computed set of detours into two subsets: the subset of *alterable* detours and the subset of *unalterable* ones. This partition is done by the owner based on semantical criteria. All the remaining edges can not be used as *synchronization edges*. Indeed, some of them may be changed by an attacker without too much altering the semantics of the data which would result in the impossibility to resynchronize during the detection process and makes the watermark ineffective. For instance, we should not use the title as synchronization key because it can be altered, e.g. by adding spaces or changing the case of some characters, without changing its semantics. Conversely, the path in the `url` is not likely to be changed in an uninvertible manner (e.g. replacing letter `'a'` by code `%61`). The corresponding edges in the automaton can be chosen as *synchronization* ones.

Example 58 *A natural choice for watermarking news items is to modify the least significant part of the date. This can be achieved by using only detours from states q_{17} to q_{20}, detours from states q_{18} to q_{21} and detours from states q_{19} to q_{21} as alterable ones.*

STEP 3: On-the-fly Watermarking. In this last step, the core of ℓ-`détour`, some portions of the stream are changed to insert the watermark. It is called `streamWatermark` and sketched on Fig. 5.3. Its execution is basically a walk on the automaton used to typecheck the stream. At each move, the last covered edges are changed if they match an alterable detour of length ℓ. Inputs of `streamWatermark` are a stream σ, the private key K_{p} of its owner and an extra parameter γ used to change the alteration rate (on average, one alterable detour out of γ is altered).

The `streamWatermark` procedure uses two variables: p and K_{s}. The path p is a finite queue having size at most ℓ containing the last covered edges, used as a finite FIFO: before adding a new edge at the end of a full p, its first edge is discarded. When p is full, it contains a candidate detour, likely to be changed if it matches an *alterable* detour. The second variable K_{s} stands for the synchronization key. It is used as a bounded-size queue of tokens. It will contain any symbol that corresponds to a synchronization edge.

The `streamWatermark` algorithm starts in **A** and regularly loops back to this cell. In **A**, we read a token from the input stream which generates a move on the automaton. The covered edge is added to p. Then, we move to cell **B**. If length(p)$< \ell$, we move back to **A**. When length(p)$= \ell$, we move to **C**. In cell **C**, we test whether p is going to be changed i.e. whether p is an alterable detour (from states i to j) and whether there is at least one another other detour from i to j. When these two conditions are met, we move to the watermark cell **E**. In **E**, the path p is converted into an integer: its rank in an arbitrary ordering of all detours from i to j. This integer, together with the synchronization key K_{s}, the private key of the owner K_{p} and γ, is passed to the procedure `intWatermark` (Alg. 6). Its output is the number of a new detour which labelling symbols will be added to the output stream. This procedure, derived from [7], uses a pseudo-random generator $\mathcal{R}$ seeded with $K_{\mathrm{s}}.K_{\mathrm{p}}$ to choose (1) whether the passed integer is going to be altered or not (2) which bit of the passed integer is going to be modified and (3) what will the new value of this bit. The synchronization key K_{s} is reseted to the empty queue. Remark that this modification only depends on the private key of the owner and tokens of the stream which are not altered. If the conditions to move to cell **E** are not met, we move to cell **D**. Path p not being an alterable detour does not mean that its suffix of length $\ell - 1$ is not the prefix of another detour. So, in **D**, the first edge of p is discarded and, if it is a synchronization edge, its labelling token c added to K_{s}. Simultaneously, c is added to the output stream. The process loops back to the initial cell **A**.

Hence, the ℓ-`détour` algorithm outputs 0,1 or ℓ tokens every time it reads a token from the input stream. If N tokens have been read from the input stream, at least $N - \ell$ and at most N tokens have been outputted which makes the process a real-time one. The output of `streamWatermark` is a stream of the form $c_1e_1c_2e_2...$ where each c_i comes from the input stream and e_i is the result of a pseudo-random choice seeded with the synchronization part of c_i concatenated with the private key of the owner. Each segment e_i has length ℓ.

Algorithm 6: `intWatermark`(i,K_{s},K_{p},γ)

Output: $1 \leq j \leq n$

1 $\mathcal{R}$.seed($K_{\mathrm{s}}.K_{\mathrm{p}}$) /* seed the random generator */ ;

// (1) decide whether i is going to be changed

2 **if** $\mathcal{R}$.*nextInt()* % $\gamma = 0$ **then**

3 $p = \mathcal{R}$.nextInt() % $\lceil \log_2(n) \rceil$ /* (2) choose which bit of i to change */ ;

4 $b = \mathcal{R}$.nextInt() % 2 /* (3) new value of bit p of i */ ;

5 $j := i$ where bit p is forced to b;

6 return j;

Example 59 *Suppose that we are in the middle of the watermarking process of the* XML *segment of Fig. 5.1. Detours of length $\ell = 1$ have been chosen and the partition of detours has been done in Example 58. Suppose also that the algorithm has just reached cell **A**, that the current position on the automaton is state q_{13} (last read token is -), that $K_{\mathrm{s}} = K_{\mathrm{s}}^0 =$ <url>http://www.imdb...</url> and $p = q_{12} \vec{\rightarrow} q_{13}$. The path $q_{12} \vec{\rightarrow} q_{13}$ has length 1 but is not a detour, so we move to cell **D** through cell **C**. In cell **D**, the first token of*

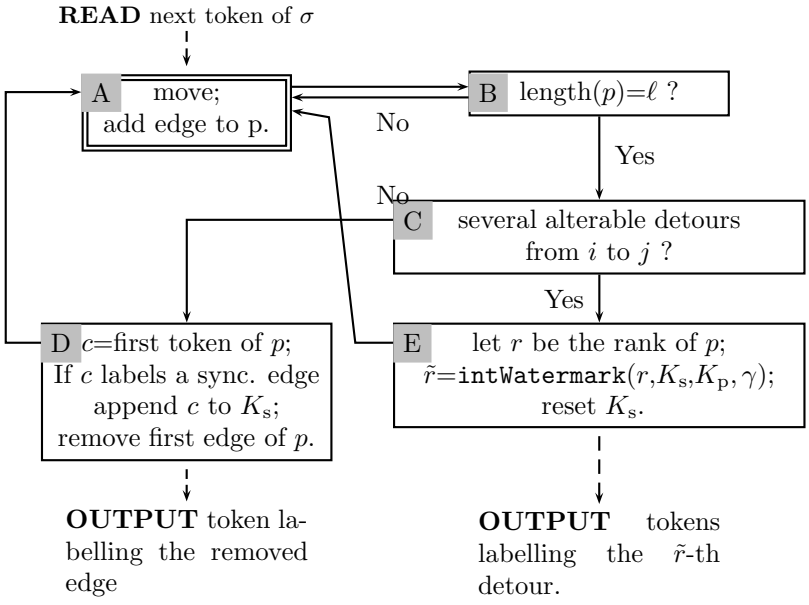


Figure 5.3: `streamWatermark`(σ,K_p,γ)

*p, `-` is removed, appended to K_s and added to the output stream. Then, $p = []$ and we move to cell **A**. The token `0` is read from the input stream and the edge $q_{13} \xrightarrow{0} q_{14}$ appended to p. Still, p is not an alterable detour and the same sequence of steps through cells **B,C,D** is performed. Then, the algorithm moves through edges $q_{14} \xrightarrow{5} q_{16}$, $q_{16} \xrightarrow{-} q_{17}$ and $q_{17} \xrightarrow{2} q_{20}$; the tokens `5,-,2` are processed the same way the token `0` was. The token `3` coding for the lowest significant digit of the day in the month is read in cell **A**. The path $p = q_{20} \xrightarrow{3} q_{21}$ is a detour of length 1 from states q_{20} to q_{21}. Since there are 10 detours between these states, we move to watermarking cell E. The `intWatermark` procedure is called with $K_\mathrm{s} = K_\mathrm{s}^0$`.05-2` and $r = 4$ (p is the fourth detour from q_{20} to q_{21}). A one-way cryptographic choice of a new detour is done by Alg. 6, depending only on K_s and K_p. For instance, if `intWatermark` outputs `7`, the seventh detour is chosen and the token `6` added to the output stream. The watermarked date is `1989-05-26`. Then, K_s and p are reseted and we loop back to **A**.*

5.2.5 Quality Preservation: Setting Alteration Frequency γ

The following theorem quantifies to what extent the quality of a watermarked stream is preserved. Let G be a stream automaton. Let S (resp. E) be the set of starting (resp. ending) nodes of the *alterable* detours. We define the *inter-detours* distance c as the length of the shortest path between a node in $E \cup q_0$ and a node in S. For the automaton of Fig. 5.2, $\{q_{17}, q_{18}, q_{19}\} \subseteq S$ and $\{q_{20}, q_{21}\} \subseteq E$ so c is at most the minimum of the distances between q_0 and q_{17} and between q_{21} and q_{17} (the actual *inter-detours* distance can not be given because of the partial specification).

Theorem 60 *Let σ^N a finite prefix of a stream and $\tilde{\sigma}^N$ its watermarked version using `ℓ-détour`. Then, at most $d(\sigma^N, \tilde{\sigma}^N) \leq (1 + \frac{c}{\ell})^{-1}$ and on average $d(\sigma^N, \tilde{\sigma}^N) \leq \frac{1}{\gamma}(1 + \frac{c}{\ell})^{-1}$.*

Proof. A finite segment σ^N of a stream σ can be written as $\sigma^N = c_1 e_1 .. c_n e_n r$ where $c_1, .., c_n$ are token sequences used as synchronization keys, $e_1, .., e_n$ are token sequences labelling detours and r is the remaining. `ℓ-détour` introduces a distortion of at most $n\ell$. Since the length of each c_i is at least c, the relative distortion $\varepsilon = \frac{n\ell}{\sum_{1,n} |c_i| + n\ell + |r|}$ is such that $\varepsilon \leq (1 + c/l)^{-1}$. On average, $\frac{1}{\gamma}$ pairs $c_i e_i$ are altered. □

Hence, for a maximum error rate $e = 0.1\%$, a detour length $\ell = 2$ and an inter-detour distance $c = 10$, the value of γ is chosen so that $\frac{1}{\gamma}(1 + \frac{c}{\ell})^{-1} \leq e$ i.e. $\gamma \approx 6000$. So, on average, one over 6000 tokens labelling *alterable* detours should be altered to comply with this error rate.

5.2.6 Watermark Detection

Since the alterations performed by the watermarking process depend only on the value of the private key K_p of the owner, exhibiting a key and making the dependencies appear is a strong proof of ownership. The detection process locates the synchronization keys and checks whether the detours taken by the suspect stream match what would be their watermarked value. It is very close from the watermarking algorithm except that the content f the stream is not changed. We use two counters, tc and mc, tc standing for *total count* and mc for *match count*. We increment tc every time we meet a detour that would be watermarked (this corresponds to line 2 of Alg. 6). We increment mc every time a detour matches what would be its watermarked value. Therefore, $tc \geq mc$. When $tc = mc$, we can conclude of the presence of a watermark. When $tc > 0, mc = 0$, we are probably in front of an attacker who successfully inverted every bit of the mark. This inversion is considered as suspicious as the full presence of the mark (think of it as a negative image of a black and white picture). For a non-watermarked stream, we can assume that there is no correlation between the distribution of the data and the pseudo-random watermark embedding process (assumption verified in our experiments). In this case, the probability that each bit of a detour matches what would be its watermarked value is $1/2$. Then, we can await for tc to be twice the value of mc when there is no mark. To sum up, the watermark is found when $|mc/tc - 1/2| > \alpha$, where α is a predefined threshold. The choice of α is very important: if α is too large, the detection raises false alarms; if α is too small, slightly altered marks become undetectable, raising false negatives. The choice of α is discussed in the next section. Remark also that only the suspect stream and the private key of the owner are needed to check for a watermark.

5.3 Robustness: Analysis and Experiments

A watermarking algorithm is said to be *robust* when an attacker, unaware of the secret key used for watermark embedding, has to alter more the data than the watermarking process did, in order to remove the mark. In that case, the attacked stream suffers a huge loss of semantics, which is very likely to destroy their quality.

5.3.1 Synchronization Attacks

A watermarked stream can be attacked by modifying synchronization parts. Indeed, ℓ-`détour` requires these parts to remain identical for detection. Such attacks are limited by the *constant requirement to keep streams valid with respect to the input type of their consumers*. A non-valid stream cannot be resold by a malicious user. As explained in **STEP 2** of ℓ-`détour`, synchronization parts are chosen to be semantically relevant which means that they cannot be changed without widely affecting its semantics. Therefore, a type breaking attack requires to alter data semantics more than the watermarking process did.

5.3.2 Detours Attacks

Since the attacker is unaware of which detours were actually altered, two strategies are available to him. First, he can try to remove the mark by randomly modifying the altered detours. We model this attack as a *random attack*.

Random Attack. For $0 < p < 1$, a random attack of parameter p is an attack inverting each bit of the watermark with a probability at most p. The false negative occurrence probability $p_{fn}(p)$ is the probability that an attacker performing a random attack of parameter p cheats the detector. Theorem 61 (see [42] for a complete proof) shows how to choose α (detection threshold) and tc (number of altered detours to poll) to get this probability maximally bounded by an owner-defined probability δ (e.g. $\delta = 10^{-6}$). These parameters also allows for a false positive occurrence probability p_{fp} bounded by δ.

Theorem 61 *Let* $0 < \delta, p < 1$, $tc \in \mathbb{N}$, $p \neq 1/2$, $tc_0 = \frac{-\log(\delta/2)}{2(1-2p)^2}$, $\alpha_1(tc) = \frac{-\log(\delta/2)}{2tc}$ *and* $\alpha_2(tc) = \frac{1}{2} - p - \sqrt{\frac{-\log(\delta/2)}{2tc}}$. *Then,*

$$(tc \geq tc_0 \text{ and } \alpha_1(tc) \leq \alpha \leq \alpha_2(tc)) \Rightarrow (p_{fp} \leq \delta \text{ and } p_{fn}(p) \leq \delta).$$

PROOF. (sketch) The fact that a detour matches its watermarked value is seen as the outcome of a Bernoulli's law of parameter 1/2. Suppose that we are able to retrieve n possibly watermarked positions in the stream. The probability that a false positive occurs is exactly the probability that the number of positive outcomes in n outcomes of Bernoulli's experiments deviates from the standard value $n/2$ by a distance $\alpha.n$. The higher n is, the smaller this probability. It can be bounded using a Hoeffding inequality [50] to obtain a maximal bound for the occurrence of a false positive. Similarly, one can bound the probability that a false negative occurs. By combining these two results, we find the minimum number of potentially watermarked detours one must consider to test the presence of a watermark and simultaneously stay under the target probability δ. □

Listen& Learn Attack. When a synchronization key is met twice, the two corresponding watermarked bits will have the same position and the same value. If $c.e_1$ and $c.e_2$ are two sequences *synchronization key.detour*, the watermarked bit is among the set of bits which have the same value in the binary representations of the rank of e_1 and the rank of e_2. An attacker may try to learn such dependencies in order to perform a *Listen & learn* attack. This attack consists in the following two steps. First, *learning* associations between synchronization keys values and watermarked bits. Second, *attacking* the watermark using this knowledge. Notice that this can not be done in constant memory and requires an external and efficient storage. This does not comply with computational constraints on streams, that also apply to the attacker.

5.3.3 Experiments

Test Sample. We used RSS news feeds provided by CNN [1] from September 8th 2005 to September 14th 2005 for a total of 1694 news items (or 523041 tokens). *Alterable* detours were chosen to be the edges associated to the highest significant digit of the minutes field and the lowest significant digit of the seconds field. Hence, dates of news item are changed by at most 50 minutes and 9 seconds. Synchronization keys include the content of the `link` element and edges not part of an alterable detour in the `pubDate` element.

Detour-witching Attack. A *detour-switching* attack consists in randomly switching *all* alterable detours. It is parameterized by the alteration frequency q: with probability $1/q$ each detour is replaced by another one, having same start and same end, randomly chosen. We performed experiments for various values of q and γ. A summary of the results is displayed in Table 5.1(a). For each combination of q and γ, the set of news items was watermarked and attacked 100 times. We count the number of positive detections **PD** of the watermark and the relative extra alteration **QL** introduced by the attack, compared to the watermarking process. If the watermarking process alters **WL** tokens of the stream, then the attack has an overall distortion of **WL**+**QL**. For instance, when $q = 1$ and $\gamma = 3$, the attack successfully erases the watermark (**PD**= 0%) but at the price of a significant quality loss **QL**= 0.39% compared to the alterations introduced by the watermarking process **WL**= 0.22%. On the contrary, for $q = 1$ and $\gamma = 1$, the attack is a success (**PD**= 0%, **QL**= 0%). This shows that choosing $\gamma = 1$ is a bad idea for the data owner, as it means watermarking *every* possible position, hence giving a severe hint to the attacker. As soon as $\gamma > 1$, the mark is not removed if the attack does not alter more the stream than the watermarking process did.

Listen & Learn Attack. We performed experiments of this attack using two strategies. In the *destructive* strategy we change every *alterable* detour unless we know it is a watermarked one. In the *surge* strategy, a detour is altered only if we are sure it is a watermarked one. We performed experiments for different learning times, *ltime* ranging from 100 detours to 1500. The detection process begins after the end of the learning period to maximize the effect of the learning attack. For each strategy and learning time combination, 100

Table 5.1: Attack Experiments: **WL** (quality loss due to watermarking), **QL** (*extra* quality loss due to attacks), **PD** (ratio of positive detections)

q \ γ	1 (high rate)	2	3 (low rate)
1	**WL**:0.65% **QL**:0% **PD**:0%	**WL**:0.32% **QL**:0.38% **PD**:0%	**WL**:0.22% **QL**:0.39% **PD**:0%
2	**WL**:0.65% **QL**:0% **PD**:100%	**WL**:0.32% **QL**:0.19% **PD**:100%	**WL**:0.22% **QL**:0.19% **PD**:100%
3	**WL**:0.65% **QL**:0% **PD**:100%	**WL**:0.32% **QL**:0.13% **PD**:100%	**WL**:0.22% **QL**:0.13% **PD**:100%

(a) Random Attack
Failure probability $\delta = 0.01$

strategy \ *ltime*	100	500	1500
surge	**QL**:0.27% **PD**:100%	**QL**:0.39% **PD**:100%	**QL**:0.24% **PD**:100%
destructive	**QL**:0.57% **PD**:52%	**QL**:0.49% **PD**:100%	**QL**:0.27% **PD**:100%

(b) Listen & Learn Attack
$\delta = 0.01$,$\gamma = 3$ and **WL**= 0.22%

experiments were performed. Results are presented in Table 5.1(b). In only one case, the watermark is removed. This is not surprising because when $ltime = 100$, the destructive strategy is a random attack with $p = 1$. Indeed, not enough knowledge has been acquired. Even for longer learning times, the attack does not affect the detection.

5.4 Related Work

Our work is an extension of [7] which considered relational database watermarking. In [7], the watermarked information is located in the least significant bits of numerical values whereas ours is located at any position, provided this position can be localized by an automaton. Type-preservation is implicit since the structure of the databases (relation name, attribute names, key constraints) is not altered. In the XML context, structure is far more flexible and can be used to embed watermarking bits. This motivates structural modifications in the purpose of watermarking, but while keeping the data usable, i.e. respecting its original type. Such structural modifications are not discussed. It is noteworthy that our automata-based model can mimic their algorithm for numerical values with a fixed size (which is a usual hypothesis in practice).

In [105], a watermarking scheme for sensor streams is proposed. Streams are defined as continuous sequences of numerical values. Watermarking is performed by altering salient points of the stream. This method can be seen as type-preserving since a flow of numerical values is mapped to another flow of numerical values. But this typing system (any sequence of numerical values) is very poor and numerical streams can not be considered as really structured. In that sense, the problematic is different from ours.

Other works [26, 39, 53, 82, 90, 121] address watermarking XML information in various contexts. In all these works XML documents are viewed as a whole, and not as streaming information. In [26, 53, 82, 121], watermark embedding values are located through the use of specific XPATH queries. It is not discussed whether these techniques can be applied in a streaming context but a starting point is that XPATH can not be efficiently evaluated over streaming data [34]. Only one work [53] considers structural modification as bandwidth for watermarking which are often viewed as attacks [90, 121] watermarkers must deal with. A theoretical work [39] explores the watermarking of XML databases while preserving constraints which are specified trough parametric queries. Type constraints does not fit into this framework.

5.5 Conclusion

In this work, we have presented the ℓ-`détour` algorithm which permits the embedding and the detection of copyright marks into XML streams. Thus, it enables detections of illegal redistributions of such objects. Future work is to study whether it is possible to detect watermarks after one or several transformations by consumers. Obviously, this is impossible in the most general setting but preliminary results [42] show that

this question can be answered for a restricted class of transformations, expressing deterministically invertible stream rewritings.

Related publications

- Julien Lafaye and David Gross-Amblard. XML Streams Watermarking. In *20th Annual IFIP WG 11.3 Working Conference on Data and Applications Security (DBSec2006)*, Sophia Antipolis, France, 7/31 - 8/02 2006, pages 74–88.

6
Symbolic musical scores

In this section we propose a new watermarking method that hides the writer's identity into symbolic musical scores featuring fingering annotations. These annotations constitute a valuable part of the symbolic representation, yet they can be slightly modified without altering the quality of the musical information. The method applies a controlled distortion of the existing fingerings so that unauthorized copies can be identified. The proposed watermarking method is robust against attacks like random fingering alterations and score cropping, and its detection does not require the original fingering, but only the suspect one. The method is general and applies to various fingering contexts and instruments.

6.1 Introduction

In this work we consider symbolic musical scores that contain *fingering annotations*. Such fingerings ease the score interpretation for the beginner player, and can guide the professional player. Producing high quality fingerings is a complex and costly task for the score writer. Up to now, it mainly remains an hand-made task, although several automatic fingering methods have been proposed recently [11, 48, 119].

The score writer's investment is threaten by the development of musical scores in digital form. Any buyer of such scores can obtain a perfect copy of the files and resell illegal copies. Watermarking is a known tool to protect the intellectual property of digital content, and it can be envisioned for musical scores as well. This would enable the distribution and sharing of score files marked by the copyright of their owner(s), just like score sheets are nowadays, but with the numerous advantages associated with the digital format.

Several methods have been proposed to hide the owner's identity into score images, by changing pixels [32], staff thickness [96] or symbols shape [97, 98]. These approaches are well fitted for protecting score images, but are not relevant for data exchange in a symbolic format like MusicXML [92]. Given the high cost of producing a symbolic digital score, writers may demand a robust mechanism to embed their copyright mark in the music symbolic representation. This copyright mark must be preserved throughout the operations that can be applied to the digital representation (e.g., transposition). It should not depend on side aspects such as graphical output details (e.g., the thickness of staff lines) which can easily be replaced or even eliminated without harm, as they are not part of the symbolic representation. Finally, the watermark should not alter the music content. In order to satisfy these requirements, our approach consists in watermarking the existing scores annotations. In the present chapter we apply the idea to fingering annotations. Up to our knowledge, this is the first work on watermarking the music semantics itself.

The key idea of the method, given a musical score and a hand-made high quality fingering, is to choose several short secret fragments of the score. Given a score fragment, we replace the existing fingering with another fingering, chosen secretly among several computer-made fingerings of comparable quality. All secret choices are made using a cryptographic pseudo-random number generator, seeded by a summary of the musical structure and with a secret key known only by the legitimate owner. The resulting fingering will be published with the musical score. Finally, given a suspect score, the correspondence of the suspect fingering with our secret choices on our secret fragments acts as the proof of ownership. Our method applies to any fingering scenario, as soon as a quality metric of fingerings is available along with an automatic fingering method for small fragments (such as in piano or guitar music for example).

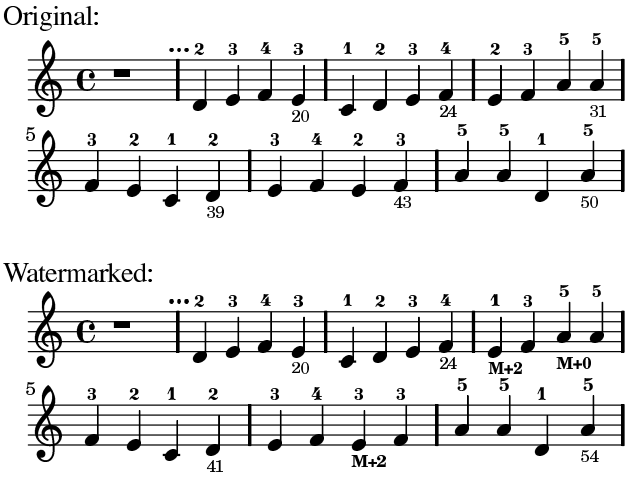


Figure 6.1: Different fingerings of the same score, with cumulative costs

It should be clear that we protect the combination of the score and its fingering, and not the score itself. We also suppose that the attacker cannot afford to alter the score significantly, as this would result in an unsellable score (nevertheless we moderate this assertion in Section 6.3).

6.2 Fingering and watermarking

6.2.1 Fingering

The method proposed in this work applies to any fingering context, but for the sake of simplicity we will focus on right-hand piano fingering for melodic inputs. Given a score in symbolic notation, we abstract it as a sequence $s = (n_1, \ldots, n_N)$ of N consecutive notes. A fingering $f(n_i)$ for a note n_i is an integer in $\{1, 2, \ldots, 5\}$, where number 1 to 5 represents the right-hand fingers, respecting the usual conventions. For example, $f(A) = 2$ means that note A will be played by the forefinger.

The watermarking method uses an estimate of the quality of a fingering, that is related to the player inner feelings. We suppose the existence of a cost function $cost(f, s)$ that provides the cost of fingering f for the score s: the higher the cost output by this function, the lower the quality of the provided fingering (such functions exist for several instruments like piano [48]). We will explicit this function in our experiments in Section 6.5, but our method applies to any such function. We also often use the cost of a fragment w of the score s, that we denote $cost(f, w, s)$.

The first staff of Figure 6.1 presents an original score fragment with fingering annotations build by the score writer. Fingering annotations appear above the score. Annotations below the score are presented here only for the purpose of explanation, but are not published by the score writer. They show the cumulative cost of playing the score with the corresponding fingering (for example, playing the whole score costs 50 according to the chosen cost function).

6.2.2 Watermarking protocols

A watermarking protocol is a pair of algorithms $(\mathcal{W}, \mathcal{D})$, where $\mathcal{W}$ and $\mathcal{D}$ are respectively the marker and detector algorithms (see Figure 6.2). Given an original score s and a high quality fingering f, the score writer will watermark it by obtaining a specific fingering $f_M = \mathcal{W}(s, f, \mathcal{K})$, depending on a secret numerical key $\mathcal{K}$. The watermarked score (s, f_M) is sold to the customer. If a suspect copy (s^*, f^*) is discovered, the detector $\mathcal{D}$ applied on (s^*, f^*) using the secret key $\mathcal{K}$ should output *guilty* if f^* was obtained from f_M, and *not guilty* if f^* is a fingering obtained independently from f_M. A watermarking protocol is said to be *blind* if the original fingering is not needed at detection time, which may be useful as writer's fingerings may

not be accessible easily or archived properly. The suspect fingering may have been also attacked/distorted before reselling, in order to erase the watermark. A watermarking protocol is said to be *robust* if it can still detect reasonably altered fingerings. Finally, respecting usual conventions, marker and detector algorithms are public, and their security relies only on the secret key.

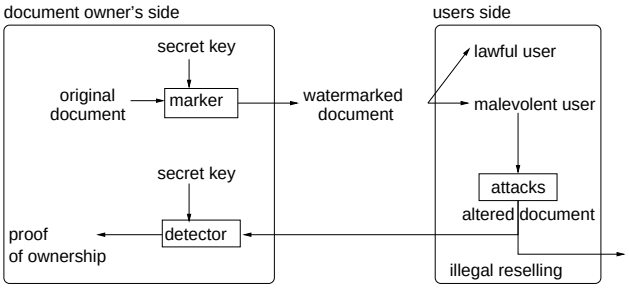


Figure 6.2: Protecting score and fingering by watermarking

6.3 Fingering Watermarking

6.3.1 Watermarking algorithm

Algorithm 7 gives the pseudo-code of the watermarking algorithm. Given a score s, the algorithm scans s by considering only a window of k consecutive notes (line 4 and 5). For each window, we first decide if it constitutes a good candidate for watermarking (line 6 and 7). This choice is secret and is based on the window content, the secret key $\mathcal{K}$ and a watermarking ratio γ known only by the score writer (this will be explained in the next section).

Algorithm 7: Watermarking

Input: a score s of N notes $n_1, \ldots, n_N$, a high quality fingering f for s, a secret key $\mathcal{K}$, a window size k, a quality threshold ε, a period γ.
Output: A watermarked fingering f'.

```
1  begin
2  |  // copy f to f'
3  |  f' := f
4  |  for i = 1 to N − k + 1 do
5  |  |  w = n_i.n_{i+1} ... n_{i+k−1} // reference window
6  |  |  seed PRNG G with signature(w).K
7  |  |  if (G.nextInt() mod γ = 0) then
8  |  |  |  // try to watermark the first note
9  |  |  |  f'(n_i) := G.nextInt() mod 5
10 |  |  |  if (|cost(f', w, s) − cost(f, w, s)| > ε) then
11 |  |  |  |  // revert changes
12 |  |  |  |  f'(n_i) := f(n_i)
13 |  return f'
14 end
```

If a given window w is considered for watermarking, we focus on its first note n_i. We try to replace the original fingering $f(n_i)$ for this note by another one, $f'(n_i)$, also chosen secretly between the 5 possible fingerings for our piano example (line 9).

We compare the cost of this new fingering $cost(f', w, s)$ on window w with the cost of the original fingering $cost(f, w, s)$ on w (line 10). If the new cost exceeds the previous one by a limit ε, we cancel this modification (line 12). If the new fingering has a reasonable cost, we keep it for publication. Parameter ε, chosen by the score writer, controls the allowed amount of alteration that results from the watermarking process, and guarantees to produce fingerings with a good quality.

The second staff on Figure 6.1 demonstrates the process. For example, the 9^{th} note (E) is considered for watermarking. Its original fingering (finger 2) has been replaced by a new fingering (finger 1). This yields an overhead cost of 2, which is considered reasonable for this example. The overall watermarking process yields a total overhead cost of 4 on the score fingering.

6.3.2 Randomness

We now explain how random choices are made. Given a window w, we compute its musical signature based its core music content (*signature()* function, line 6). The signature is independent from annotations and ornaments that are pointless for our algorithm. It is robust against naïve transposition attacks as it transposes the score into a common key (but of course, fingering costs are computed according to the original score). It is also invariant against score rewriting replacing a note or group of notes by an equivalent encoding (for example, replacing a half note by two tied quarters). In this work, the signature is the concatenation of transposed note pitches, where consecutive equal pitches are suppressed. For example, the signature of `ABAABC` is `ABABC` (seen as a number), and time is not taken into account.

We concatenate this signature with the secret key $\mathcal{K}$ (a number), known only by the score writer. Then, we seed a cryptographic pseudo-random number generator (PRNG) with this number (as in [7]). This generator is used for all subsequent choices and has interesting properties. First, if it is seeded with the same value, the produced numbers are deterministic. Hence, if we know the secret key, we will be able to reproduce the pseudo-random choices made at watermarking time. Second, if the secret key is unknown, the generator outputs look completely random and can not be reproduced. Hence an attacker, unaware of the secret key, is fighting against randomness.

6.3.3 Detection algorithm

The detection algorithm (see Algorithm 8 for the pseudo-code) proceeds like the marker algorithm. Using the same window size, watermarking ratio and secret key used at watermarking time, we seed the generator with each window signature and the secret key (line 7). Hence, the same random choices made at watermarking time are reproduced. Thus we can locate exactly those windows selected at watermarking time (line 8). Then, *since the detector does not have the watermarked fingering for comparison* (blind detector), we have to assess that this position has really been used for watermarking. For that, we replace the fingering of the first note by the awaited one, using the random generator (line 11). We then compute the cost of this fingering. If it exceeds the error limit ε, we discard this window and restore the initial fingering (line 20). If error limit is respected, then this position is probably a watermark (line 14). We then compare the awaited fingering with the found one (line 15). For the whole score, we maintain the ratio of the number of matching fingerings with the number of windows considered for detection. If this ratio exceeds a given threshold (line 24), we consider the score as suspect (the threshold value is discussed below).

6.4 Discussion

In this section we discuss several classical issues related to watermarking algorithms.

Impact on quality. Since the PRNG outputs random numbers with uniform distribution, the probability for a window w to be considered for watermarking is $1/\gamma$. The impact of watermarking this window can not be higher than ε. Hence, for a N notes score, the mean overall alteration is at most $\varepsilon\lfloor N - k \rfloor/\gamma$.

Algorithm 8: Detection

Input: a suspect score s of N notes $n_1, \ldots, n_N$ with its fingering f^*, a secret key $\mathcal{K}$, a window size k, a quality threshold ε, a period γ, a security parameter δ.

Output: *guilty* or *not guilty*.

```
1  begin
2  | // copy f* to f'
3  | f' := f*
4  | total := 0, match := 0
5  | for i := 1 to N − k + 1 do
6  | | w = n_i.n_{i+1} ... n_{i+k−1} // reference window
7  | | seed PRNG G with signature(w).K
8  | | if (G.nextInt() mod γ = 0) then
9  | | | // check this window
10 | | | // compute awaited value
11 | | | f'(n_i) := G.nextInt() mod 5
12 | | | if (|cost(f', w, s) − cost(f*, w, s)| ≤ ε) then
13 | | | | // probably watermarked position
14 | | | | total++
15 | | | | if (f'(n_i) = f*(n_i)) then
16 | | | | | match++
17 | | | else
18 | | | | f'(n_i) := f*(n_i) // revert changes
19 | if (match/total > 1/5 + threshold(N, δ)) then
20 | | return guilty
21 | else
22 | | return not guilty
23 end
```

Window size. As the window size k increases, the amount of randomness injected into the random generator extends. If we consider reasonable scores whose notes spans 2 octaves, there is up to 14^k potential fingerings for k consecutive notes. We chose $k = 5$ in our experiments, leading to half-a-million distinct window signatures.

False positives probability and threshold function. A false-positive detection occurs when the detector considers a random score as guilty. Clearly, this probability must be negligible. Let δ be this acceptable probability, say $\delta = 10^{-10}$. Let us consider a random score. The probability of a given window to be selected by the detector is $1/\gamma$. For piano fingering, the probability of a fingering to correspond – by chance – to the watermarked one is $1/5$ (as there is 5 different possible fingerings). Hence the average number of total matches on a random score is $\lfloor N - k \rfloor / 5\gamma$. By the Hoeffding Bound [50], the probability that the detector ratio $\frac{match}{total}$ on a random score deviates from the previous average is such that

$$P[|\frac{match}{total} - \frac{1}{5}| > threshold(N, \delta)] < e^{-2\frac{N}{\gamma} threshold(N,\delta)^2}.$$

Hence, choosing $threshold(N, \delta) = \sqrt{\frac{\gamma}{N} \ln \frac{1}{\delta}}$ guarantees a false positive rate smaller than δ. For example, on a score of 10 000 notes with a watermarking period $\gamma = 10$ and $\delta = 10^{-10}$, the recommended threshold is 0.22.

Available bandwidth. Robustness and significance are proportional to the amount of watermark bit that can be hidden. In popular guitar pieces (e.g., guitar scores and tablatures for beginners), a significant number of watermark positions are available. But music for expert players may contain only a few fingering annotations. If this number is not sufficient to reach the security limit, or if the musical corpus is made of small pieces only, a natural extension is to consider the watermarking of an entire piece collection (collected in a CD for example). The watermark is spread on the collection, and since the detection method uses only a finite-size sliding window, the order of pieces within the collection is pointless at detection time. The method is also robust enough to recover the watermark on a subset/superset of scores.

Attacks. An attacker suspecting the occurrence of a watermark may try to evade detection by several means. First, the attacker can add easy-to-correct errors in the fingering. To be successful, the attacker will have to add such errors all along the piece, in order to erase sufficient watermark positions. Hence the overall fingering is full of errors. Second, the attacker can leave the fingering unchanged, but add errors on the score itself, in order to break synchronization with the fingering. If errors are simply note rewritings, the signature method will probably recover the correct one. If the error is big, it will break one watermark position. Again, errors must span the whole score to be efficient, which is unreasonable (due to lack of space, we omit the mathematical proof of these statements. They are similar to the false-positive analysis).

Another approach for the attacker is to refinger the score. A complete rewriting represents a significant amount of work, so why would this attacker bother buying a fingered score in the first place ? On the contrary, a small refingering acts as a random attack, as the attacker has no idea where to perform this fingering.

Finally, the malevolent user can attack the score structure. Brute-force transposition is not sufficient, as we normalize the score in a specific key for detection. A first technique is to resell only subscores (excerpts). This can occur even for a normal buyer using the score. However, as long as a significant fraction of the piece is present, the watermark can be detected (this fraction is typically 30% in the database watermarking literature [7]). If less than 1/3 of the piece is stolen, the loss of property is harmless. If an attacker mixes a watermarked collection with a huge number of unwatermarked pieces, the argument is similar.

A last technique is to fold or unfold the score according to repetition symbols. This attack can be counterfeited by discarding repeated parts in the *signature*() function, both for watermarking and detection.

6.5 Experiments

6.5.1 Data, cost function, parameters

Our experiments are based on 50 Chopin piano pieces from the KernScores repository [94], for a total of around 10,000 notes. Original fingerings were found with a Dijkstra algorithm using a fingering cost function close to [48] and [11] (our method supposes hand-made high quality fingerings, but this approach is sufficient to measure the watermarking impact on quality). These models encompass the cost of playing a note with a given hand position (vertical cost $cost_v(f, n)$), and the cost of the transition between one hand position to the next one (horizontal cost $cost_h(f_i, n_i \to f_{i+1}, n_{i+1})$). These costs are constant values that agree with the human hand physical possibilities (the precise definition of these costs in not relevant for the present work, we refer the reader to [11] for in-depth explanation.) The cost $cost(f, n)$ of a fingering f is the sum of its horizontal and vertical costs, i.e.,

$$cost(f, n) = \sum_{i=1}^{N} cost_v(f_i, n_i) + cost_h(f_i, n_i \to f_{i+1}, n_{i+1}).$$

We used window size $k = 5$, error tolerance $\varepsilon = 10$ and detection threshold 0.8 (vertical and horizontal costs for one note or transition spans between 0 and +14).

6.5.2 Experiments

Figure 6.3 shows the impact of the watermarking method for various values of watermarking period γ. Clearly, a period smaller than 5 yields a huge distortion, and greater values tend toward a constant error with respect to the original fingering. Figure 6.4 and 6.5 study the impact of a random attack that tries to erase the watermark as follows: a note fingering is chosen with probability $1/\gamma_a$, and changed into a random fingering up to a cost impact of 10. Figure 6.4 shows the attack impact on the watermarked fingering quality for various values of γ_a. It appears that the attack impact is larger than the watermark impact on the fingering cost: choosing $\gamma_a < 5$ leads to fingerings with poor (unsellable) quality. Figure 6.5 shows the attack impact on the detector ratio. Choosing a detection threshold of 0.8 guarantees that all suspect fingerings are correctly detected, expect for those with attack γ_a smaller than 6. Hence, Figure 6.4 and 6.5 argue that any attack tricking the detector also destroys the fingering quality. Finally, Figure 6.6 shows that using a random secret key does not yield false positive detection (the correct key is presented at index 50.)

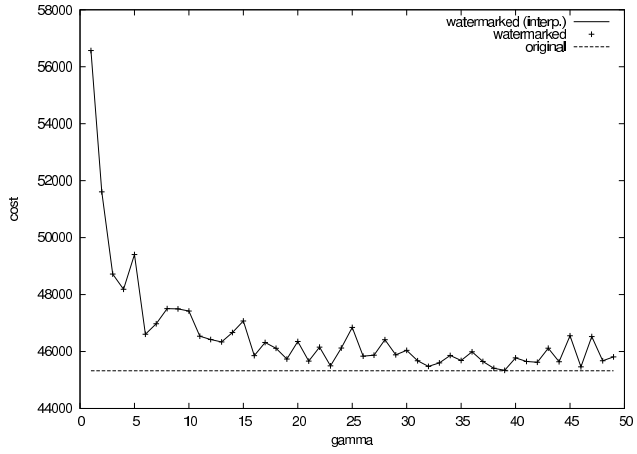


Figure 6.3: Impact of watermarking on fingering cost

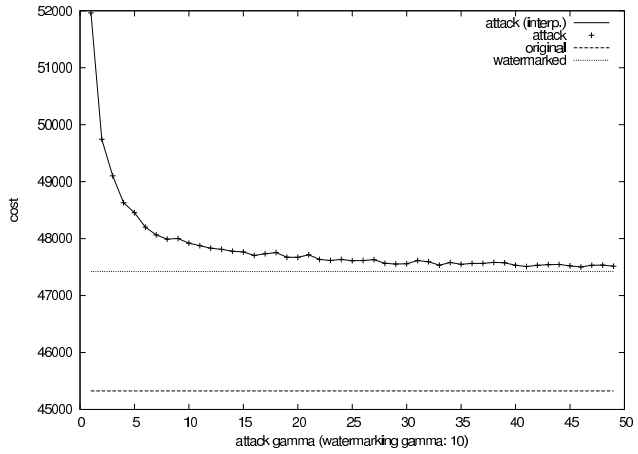


Figure 6.4: Impact of attack on fingering cost

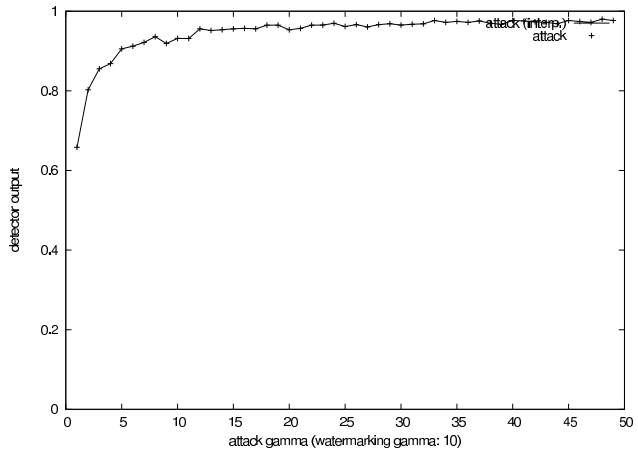


Figure 6.5: Impact of attack on detector's output

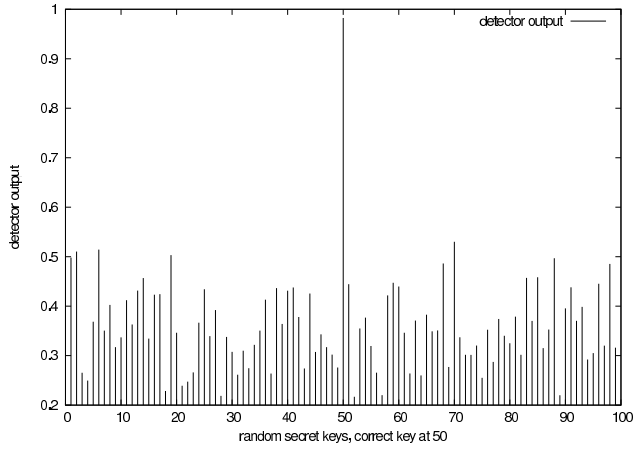


Figure 6.6: Detector output for random secret keys (correct one at 50)

6.6 Related work

Hiding information (for various purposes) in musical scores is an old story. A study of music score watermarking was performed during the WEDELMUSIC project. A good survey [79] recalls these approaches. In the visual domain, classical but adapted image watermarking techniques can be applied on the image of a musical score. The watermark can be hidden by altering grayscales, or the binary representation of images, or the pixels themselves. In the musical notation (but still into the score image), one can alter the staff thickness, the vertical or horizontal distance between notes or groups of notes, notes orientation, thickness [96] or shape [97, 98]. Little is known on information hiding into the music semantics, where our work stands.

Our method shares some similarities with database watermarking methods: watermarking of relational databases of numerical values [7], numerical data streams [106] and XML streams [66]. All these methods use the same PRNG technique, and [104] and [66] also use a finite window to scan a numerical or textual stream. The main difference is that our method has to control a non-local cost on data and may require rollbacks.

6.7 Conclusion

On-line distribution of musical scores is a promising area. Among other advantages, it could offer instant access to music collections, a wide diffusion of rare musical pieces, and computer-based services to browse, recommend, search and analyze music. However, producing music scores is a costly process and the protection of score writers against illegal copies is a prerequisite for on-line collection to emerge. In the present chapter, we propose a watermarking algorithm based on the idea that the owner signature should be based on the musical content (which can hardly be modified) and hidden in a valuable annotation of this content – namely, fingerings. We propose a simple algorithm and show that it results in an effective protection. Although currently limited to fingerings, we believe that our approach can be extended to music annotations in general, for instance texts in vocal music. We are currently investigating this larger context.

Related publications

- **David Gross-Amblard, Philippe Rigaux, Lylia Abrouk and Nadine Cullot. Fingering watermarking in symbolic digital scores. In *International Conference on Music Information Retrieval (ISMIR)*, 2009, Kobe, Japan.**

7 Geographical data

In this chapter we focus on the watermarking of geographical data in a fully applicative perspective. We put the emphasis on the building layer of common geographical data sets, and take into account both data accuracy and a so-called angular quality of the data.

7.1 Introduction

Geographical Information Systems (GIS) have existed for more than 40 years, and their application domain is now wide, ranging from environmental surveillance by country agencies to localization-aware services for individual mobile users. This phenomenon is stressed for the general public by the increasing availability of GPS devices (e.g. car navigation) and the recent development of Google Earth™ and GeoPortail™ [2]. Most of these geographical applications rely on an underlying vectorial spatial database (points, polylines and polygons). Even in Google Earth™ or GeoPortail™, where the user interface is image-oriented, the provided satellite images are semantically underlined with polygonal structures representing points of interest (viewpoint, services), buildings or road networks.

Gathering such accurate information is an onerous task for the data owner. Hence, huge and detailed vectorial databases carry a high scientific and/or economical value. For example, 50 USD is the usual fee to be licensed to use polygons from a narrow one square kilometer area. This price is 10 times higher for a reproduction license and far much higher for a full commercial license. Due to the ease of reproduction of digital media, unauthorized copy and use threaten geographical data providers. Protecting the intellectual property (IP) of rights owner is then a requirement.

On the legal side, data providers restrict the way buyers are allowed to use their data. On the technical side, robust watermarking is a known technique for IP protection. It consists of hiding a copyright mark within the data set. Embedded marks must be robust against removal attempts to be useful. In this section, we propose a robust watermarking method for polygonal data sets.

To embed the watermark, the data has to be altered. What might sounds as a drawback is common to most watermarking methods [57]. There is a trade-off between watermark robustness and data alteration: the more alterations are allowed, the more robust the embedded watermark is. So, defining precisely what makes the value of a data set is a prerequisite for watermarking.

Some applications do not rely only on spatial accuracy (i.e. the distance between a point in the real world and this point in the data set). For example, spatial accuracy is not crucial for tourist city maps designers who apply strong transformations to road polylines and building polygons in order to increase readability. Some others focus on objects like forests, cliffs and shallows for which precise borders can be difficult to define. But most applications rely on accurate data for automatic operations (e.g. service proximity search, GPS navigation, spatial analysis of risks, etc.). Accuracy can even be mandatory, e.g. for reefs locations on IHO/SHOM boat maps [111]. Finally, accurate data sets must conform with some standard reference system for interoperability purposes (e.g. the World Geodetic System – WGS84, which is the GPS reference system). So any watermarking method must respect data accuracy.

Beside accuracy, real world requirements entail specific constraints within the data set. For example, it turns out that most of the vectorial content of geographical databases consists of building polygons (80% on

the professional data set used in the experiments). Building polygons are under the scope of the *squaring* constraint: data is systematically corrected so that buildings with right angles are mapped to polygons with right angles in the data set. This *squaring* operation, available in any GIS, is systematically performed by data owners *and* data users, and increases the angular quality of the data set. It is also very invasive since potentially each point of the data set is moved. Experiments show that it also tends to increase data accuracy. But surprisingly, squaring impact has never been taken into account by existing watermarking proposals e.g. [85, 100]:

- On the owner side, watermarking is likely to turn squared shapes into skewed ones, reducing the data quality;
- On the user side, squaring is likely to wipe out the watermark of the owner, lowering its security. This natural transformation, along with other common filters, can be interpreted as an attack on the watermark.

To take these effects into account, we model the quality of a data set by means of (1) its accuracy and (2) its angular quality. This choice is motivated by a recently published survey [83], where it is deeply discussed that quality encompasses accuracy and must take into account shapes and topology.

In this section, we propose an effective method for watermarking the building layer of a GIS. This watermarking is robust against geographical transformations (including squaring and simplification) and attacks by malicious users. As far as we know, this is the first method which takes into account the essential squaring transformation. It provides a high level of security while controlling the impact on the quality of the data set (point accuracy and angular quality) and not introducing topological errors (overlapping polygons). An extended version of our method can even resists the MBR attack, which replaces each building by its minimum bounding rectangle. Moreover the scheme is blind: the original data set is not required for detection, definitely an important property for huge data sets.

A classical skeleton [7] of databases watermarking algorithms is to create a secret dependency between (1) a robust identifier of the data and (2) one of its characteristics, e.g. between the primary key of a tuple and one of its numerical attributes. Revealing this dependency acts as a proof of ownership. In our approach, we get rid of the primary key by constructing a robust identifier for each building using a well chosen portion of the highest significant bits of the coordinates of its centroid. Then, we rely on the observation that buildings have an intrinsic orientation and that most of their edges are parallel or perpendicular to this orientation. To hide a watermark bit, we expand or shrink buildings along their orientation. The expansion ratio is deterministically chosen among a set of quantized values according to the robust identifier of the polygon, the secret key of the owner and the bit to be embedded. By embedding the watermark within the shapes of a building rather that within the coordinates of its vertices, we achieve robustness of watermarks against squaring. Our scheme is also robust against other transformations we present later. Any malicious attacker has to tremendously reduce accuracy and/or angular quality of the data set to erase the watermark.

Outline After describing a simple model for the quality of a buildings database, we introduce watermarking basics and common geographical filters in Section 7.2. Our watermarking procedure is described in Section 7.3. Correction, efficiency and robustness of the method are assessed in Section 7.4, through an extensive series of experiments. Related work is exposed in Section 7.5 and Section 7.6 concludes.

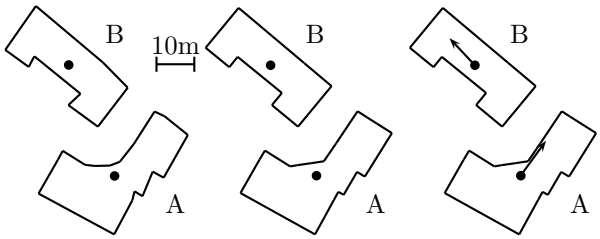
7.2 Preliminaries

7.2.1 Quality of Geographical Data

We suppose, as in any geographical application, that a reference system R_0 has been chosen and that all spatial coordinates are expressed in R_0 (e.g. cartesian coordinates on the World Geodetic System, or WGS84).

A point $p = (x, y)$ is defined by its 2-dimension coordinates (x, y) in some reference system R_0. A simple polygon $P = (p_1, \ldots, p_n, p_{n+1} = p_1)$ is represented by the list of its points. Two polygons taken from a real

data set are shown on Fig. 7.1(a). A geographical database instance is defined by (R, DB) where R is a reference system and $DB = \{P_i\}$, $i \in \{1, \ldots, N\}$ is a set of N polygons. It is always provided with some reference system otherwise it is of no use for automatic operations (nevertheless, we discuss in Section 7.4.4 the problem of missing reference system).



(a) Raw data (b) Squared data (c) Orientation

Figure 7.1: Buildings polygons

We do not rely on the order of polygons within the data set, nor on the order of points within a polygon. Furthermore, there is no primary key identifying these polygons. Polygons are supposed non-overlapping as in many geographical applications. They can have holes. In that case, we process them as the full polygon having the same envelope. More sophisticated models of spatial data expressing topology exist, but we omit these enhancements for the sake of simplicity.

The (economical) value of a data set (R, DB) is correlated with its *mean accuracy*, its *maximum accuracy* and its *angular quality*. The *mean accuracy* (resp. *maximum accuracy*) is the mean (resp. maximum) value of the distance between a point of a (real) building and its corresponding point in the data set. The *angular quality* [10] is defined as the opposite of its *angular energy*. The energy of an angle is a continuous piecewise quadratic whose minima are reached for multiples of $\pi/4$. The angular energy of a polygon is the sum of the energies of its angles. The intuition is that angles of real-world buildings are mostly right, or at least multiples of $\pi/4$. So, regular buildings have lower energy levels.

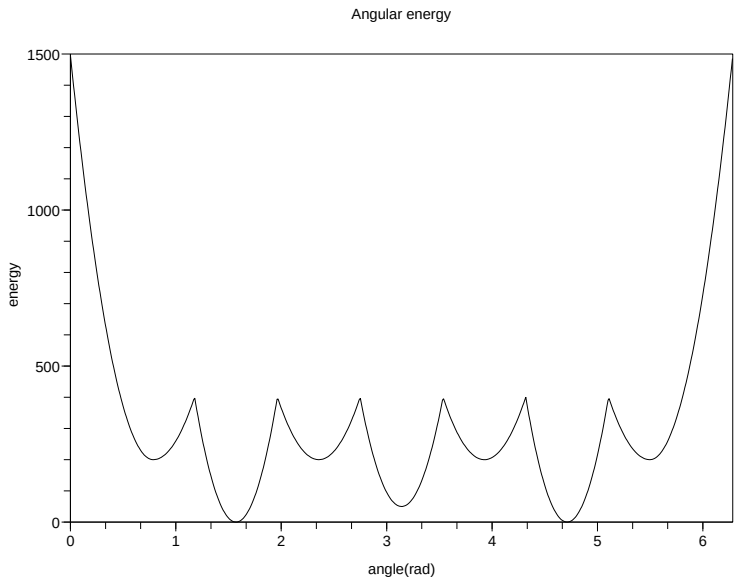


Figure 7.2: Minima of angular energy ($k\pi/2$)

7.2.2 Watermarking

A watermarking procedure is defined as a pair of algorithms $(\mathcal{W}, \mathcal{D})$, where $\mathcal{W}$ is the watermarking algorithm, and $\mathcal{D}$ is the detection algorithm. Algorithm $\mathcal{W}$ takes as inputs a data set (R, DB) , a secret key $\mathcal{K}$, tuning parameters, and produces a watermarked data set $(R, DB_{\mathcal{K}})$. The aim of the detector is, given a suspect data set (R', DB') and the secret key $\mathcal{K}$, to decide whether this data set holds a watermark or not. A watermarking procedure is said to be *blind* if the original data set is not needed by the detector $\mathcal{D}$. It is said to be *robust* if it detects marks in altered watermarked data sets. It is well known that any robust watermarking method must alter the data [57]. Hence, there is a trade-off between the allowed alteration, i.e. the allowed impact on the quality, and the robustness of the algorithm.

To evade detection, an attacker may use one of the following attacks: random alteration of point positions, mixing polygons from various data sets, applying the same or another watermarking algorithm, and specific attacks like polygon-wise rotations. We discuss these attacks in Section 7.4. Of course, the attacker, who still wants to re-sell a valuable data set must adopt a common reference system and limit the quality loss so that profit can still be made from the attacked database.

7.2.3 Geographical Filters & Attacks

Geographical data sets are likely to undergo transformations by legitimate or malicious users. A broad collection of such transformations is presented below. They can be divided into correction filters (SQ, DP), readability improvements (ETR, MBR, CE) and malicious attacks (GN, OW, CA). Nevertheless, this taxonomy is not fixed as a malicious user might apply correction filters and a legitimate one might crop a large data set to keep only the part useful to him. A robust enough watermarking algorithm should resist all of them:

- **Squaring (SQ)** For each polygon, its vertices are moved so that its angular energy is lowered. The strength of the squaring is controlled by the maximum allowed alteration d on coordinates. Fig. 7.1(b) shows a squared building, to be compared with its original counterpart depicted in Fig. 7.1(a).
- **Douglas-Peucker simplification (DP)** The Douglas-Peucker simplification algorithm [29] is a polyline simplification algorithm. It works by removing the vertices of polygons that draw small artifacts on the edges of this polygon. Its strength is controlled by a threshold distance d. The higher d, the larger the removed artifacts are.
- **Cropping (CA)** Polygons not contained within a given rectangle are discarded.
- **Gaussian noise (GN)** A random noise is added to each point of the database. The distribution of the noise has mean 0 and a variable deviation d.
- **Over-watermarking (OW)** Applying the watermarking algorithm with a different key on an already marked data set.
- **Enlarge to rectangle (ETR)** This filter replaces buildings by their bounding rectangle. Two modes are available. The first one replaces each building with a rectangle having the same surface. The second one takes as input a target scale and replaces the buildings that are too small (for a legally fixed threshold value) to be legible on a map at that scale.
- **Change elongation (CE)** Applies a fixed ratio elongation along their orientation on all buildings of the data set.
- **Minimum bounding rectangle (MBR)** Replaces each building by its minimal bounding rectangle.

Translating or rotating the whole data set are not an issue, since we focus on data sets with a reference system: such transformations can be easily reversed. Observe that printing/scanning a map drawn from the data set is out of the scope of our approach, since such paper maps are of no use for automatic operations.

7.3 Building Watermarking

7.3.1 Outline of the Algorithm

The rationale for many watermarking algorithms is to hide a secret dependency between (1) a robust part of the data set, that will survive most alterations, and (2) one of its characteristics, whose alteration is allowed up to a reasonable limit. Revealing this secret dependency acts as a proof of ownership. We build a robust identifier id_i for each polygon P_i by using the highest significant bits of the coordinates of its centroid, expressed in the predefined reference system R_0. This identifier is robust since it is invariant through the modifications of vertex coordinates, involving only least significant bits. High amplitude modifications are likely to break the identifiers but also to lead to visible shapes alterations and/or polygon overlappings. Furthermore, if the coordinates of the polygon of the centroid are expressed in a reference system R', different from R_0, it is easy to convert them back into R_0. Indeed, no geographical data comes without a reference system.

In order to hide a bit of information in polygon P_i, we expand or shrink it along its orientation. This orientation is computed relatively to the centroid (see Fig. 7.1(c)), and represents the majority weighted angle among edges directions. We present its computation in Section 7.3.3. For a rectangular shape, this orientation is parallel to the longest edge. Choosing to expand along this orientation offers several advantages. First, we observed that most edges of a polygon are parallel or perpendicular to this orientation. For example, there are 3 directions in polygon A (Fig. 7.1(c)): SW-NE, SE-NW and W-E. The main direction, i.e. the orientation is clearly SW-NE since the longest edges are heading this direction. Other directions are perpendicular or make a $\pi/4$ angle with the orientation. When a polygon is expanded along its orientation, geometrical relations between directions do not change. Second, an expansion along the orientation can still be detected if the polygon is rotated. Finally, the impact on polygon surfaces has tighter bounds compared to the case where shrinking or expanding along several directions is allowed.

It remains to compute the expansion factor to apply, and to choose which polygons are going to be altered. These operations must be done so that any attacker, aware of the watermarking method, is unable to guess on which polygons they were actually applied. A classical method to achieve this [7] is the following: use the concatenation of the given identifier id_i of a polygon and the secret key $\mathcal{K}$ of the owner to seed a pseudo-random number generator (PRNG). Use pseudo-random drawings from the generator to determine whether the current polygon is modified and, eventually, with which expansion factor. The sequence of numbers produced by the generator is predictable if and only if $id_i.\mathcal{K}$ is known. It appears purely random to anyone who does not possess this seed (an attacker may easily compute id_i, but $\mathcal{K}$ remains unknown).

Example 62 *An example of our watermarking method applied on polygons A and B is shown on Fig. 7.3. Original shapes are shown in black and watermarked ones in gray. First, we compute the centroid of A and B, obtaining for example $O_A = (293, 155)$ and $O_B = (171, 447)$. To form unique identifiers id_A and id_B, we concatenate the two highest significant digits of each coordinate, obtaining $id_A = 2915$ and $id_B = 1744$. Choosing these two digits is correct under the hypothesis that any reasonable alteration is below 10 meters and that the typical distance between any two buildings is more than 10 meters (this example considers decimal base while our algorithm considers binary base). Second, based on the pseudo-random choices of a generator seeded with id_A and the secret key $\mathcal{K}$, we decide that A must be watermarked with a mark bit 0. We compute the main orientation $\vec{u}$ of A and find the vertex p such that $\vec{u}.\vec{Op}$ is maximal. Let x_{max} denote this value. Finally, we expand the building along its main orientation so that x_{max} becomes a predefined value x^0_{max}, encoding bit 0. Polygon B is processed identically. Remark that A has been expanded whereas polygon B has been shrinked, and that most angles are invariant under this transformation.*

In the following, we detail the three consecutive steps of our algorithm: (1) computation of polygon identifiers and orientations, (2) computation of expansion factors and (3) watermarking by expansion.

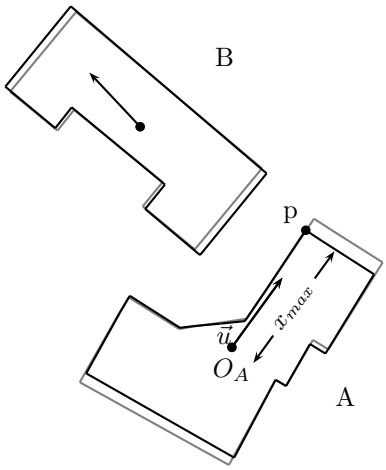


Figure 7.3: Bit embedding by expansion

7.3.2 Computing Robust Identifiers

As a robust identifier, we use the highest significant bits of the centroid of the polygon. If P is a polygon with n points $p_1, \ldots, p_n, p_{n+1} = p_1$, its area A and its centroid $O = (x_0, y_0)$ can be computed using the following common formulae:

$$A = \frac{1}{2} \sum_{1 \leq i \leq n} (x_i y_{i+1} - x_{i+1} y_i),$$

$$x_O = \frac{1}{6A} \sum_{1 \leq i \leq n} (x_i + x_{i+1})(x_i y_{i+1} - x_{i+1} y_i),$$

$$y_O = \frac{1}{6A} \sum_{1 \leq i \leq n} (y_i + y_{i+1})(x_i y_{i+1} - x_{i+1} y_i).$$

Centroids of polygons A and B are represented as black dots on Fig. 7.1(a) and 7.1(b).

We need to ensure that the chosen highest significant bits are significant enough. Suppose that the h-th bit is the least highest significant bit. On the one hand, h must be high enough to that small modifications of the polygon do not change the identifier. On the other hand, h must be small enough so that two adjacent polygons do not share the same identifier. The identifier of a polygon P is computed by pruning in the binary representations of its x and y coordinates the bits that represent powers of two at most $h - 1$ and concatenating them. We denote by $hsb(O, h)$ this operation.

$$id = hsb(O, h) = concat(hsb(x_O, h), hsb(y_O, h)).$$

As we will see later, two watermarked polygons sharing an identifier will be expanded using the same quantization step. Such an information may be used by an attacker to break the watermark. Therefore, shared identifiers must be avoided. A good choice is to select the smallest value h for which the identifiers vary from one polygon to the other. We illustrate how we compute h by means of an example. For our real-life data set (detailed in Section 7.4), we measured, for each polygon, the distances between the centroid and the farthest vertex from the centroid. We obtain a mean of 12 meters and a standard deviation of 10.5 meters, that is, most polygons have an interior maximum distance between $2(12 - 10.5) = 3$ meters and $2(12 + 10.5) = 50$ meters. Considering 3 meters as a minimum size for a polygon, most polygons have their centroid spaced by at least 6 meters. Even if there are pathological configurations, it is very likely that, by choosing $h = 2$ meters, two polygons do not share the same identifier. When $h = 2$, $l = 2^h = 2^2 = 4$ meters is considered as the minimum significant distance. Experimentally, we verified that, by choosing $h = 0, 2, 4$ and 8, we had $0, 0, 40$ and 300 cases of identifiers collision out of 4278 polygons. Hence, it seems that a good

Table 7.1: Angles Buckets

bucket	0	1	2	3	4	5	6	7	8	9
#edges	3	0	0	8	0	0	0	0	6	0
weight	9.02	0	0	62.4	0	0	0	0	45.2	0

heuristic is to use $h = \lfloor \log_2(2.(\bar{l} + \delta l)) \rfloor$ if $\bar{l}$ and δl are the mean and the deviation of the main lengths of the polygons in the data set.

7.3.3 Computing Polygon Orientation

We define the main orientation $\vec{u}$ of a polygon as the maximum weighted orientation of its edges. For instance, if e_1 and e_2 are the only edges to have orientation α, then the weight of angle α is the sum of the lengths of e_1 and e_2. The problem is that parallel walls in the real world are not necessarily mapped to parallel edges in the data set. So, we need to sum the lengths of edges that are almost parallel. We define ε the tolerance angle, that is e_1 and e_2 are considered as having the same orientation if their orientations α_1 and α_2 are such that $|\alpha_1 - \alpha_2| < \varepsilon$. To efficiently compute the orientation, we defined a bucket-based classifying algorithm based on the observation that there is often only a small number of different orientations per polygon. The algorithm consists of the following three steps. First, we create a set of k empty buckets, provided we choose k such that $\pi/k < \varepsilon$. In bucket i, we put all edges having an orientation between $(i-1).\frac{\pi}{k}$ and $i.\frac{\pi}{k}$. Hence, in buckets i and $i+1$ we have all edges that are almost equal to $i.\pi/k$. Then, we aggregate these small buckets into bigger ones by merging two buckets if there is no empty bucket between them. The main orientation of a building is computed as the mean value of the bucket having the highest cost (the cost of a bucket being defined as the sum of the lengths of the edges in that bucket). It can happen that three or more buckets need to be aggregated, leading to consider orientations as equal when their difference is greater than angle tolerance. This is very unlikely. Indeed, we observed that on buildings, there is only a few directions per polygon (2, 3 in most cases) which are clearly separated. A similar approach was followed in [30] with the main difference that this latter method requires to compute the weight of all π/k orientations and select the one with the highest weight. Our method is more efficient but may be less accurate in a restricted number of situations.

Example 63 *We illustrate the orientation computation algorithm on polygon A of Fig. 7.1(b). The number of classes is set up to* 10*. Table 7.1 contains the number of edges and the weight of each bucket. The highest cost bucket is bucket* 4*, i.e. the orientation is between* $3\pi/10$ *and* $4\pi/10 = 2\pi/5$*. The computation of the weighted mean angle of bucket* 4 *gives* 0.96 *rad.*

For some specific shapes, e.g. perfect squares and circles, our definition of orientation is ambiguous. Instead of arbitrarily choosing an orientation, we simple ignore these unusual cases for polygon expansion. On the test sample of our experiments, we had to ignore 1 polygon out of 4278.

7.3.4 Expansion as a Bit Embedding Method

In this subsection we show how to embed a single watermark bit b into a polygon P. To ensure that the watermark is robust enough, we alter the overall shape of the polygon. More precisely, we alter the longest distance x_{max} (see Fig. 7.3) along the orientation $\vec{u}$ from the centroid O to a vertex p. For a rectangular polygon, this length is half the length of the longest edge. We name *main length* this longest distance. But only altering the coordinates of p is not sufficient because it may lower angular quality (right angles may be flattened by this transformation). Hence, we choose to alter all lengths along the orientation $\vec{u}$ so that most angles are preserved. Defining by $\vec{v}$ the unary vector such that $(0, \vec{u}, \vec{v})$ is a direct orthonormal basis, watermarking is done as follows:

- compute the x coordinate x_i of each point p_i of the polygon in $(0, \vec{u}, \vec{v})$;

- compute the main length $x_{max} = \max_i\{|x_i|\}$;
- expand all points coordinates along direction $\vec{u}$ so that x_{max} becomes one of the values $\{x^0_{max}, x^1_{max}\}$ coding a watermark bit 0 or 1.

This latter operation on x_{max} is known as quantization. Given a quantization step d, we define 0-quantizers (resp. 1-quantizers) as $q^k_0 = k.d$ (resp. $q^k_1 = k.d + d/2$), $k \in \mathbb{Z}$. Intuitively, 0-quantizers (resp. 1-quantizers) are used to code a bit 0 (resp. a bit 1). To quantize the value x_{max} using the i-quantizers ($i \in \{0, 1\}$), we look for k_0 such that $|q^i_{k_0} - x_{max}|$ is minimal. More precisely, this is achieved with the following steps (quantization on 0-quantizers is presented):

- compute $k_r = x_{max}/d$;
- round k_r to the closest integer k_0;
- define the quantized version of x_{max} as $x'_{max} = k_0.d$.

The quantization process is illustrated on Fig. 7.4.

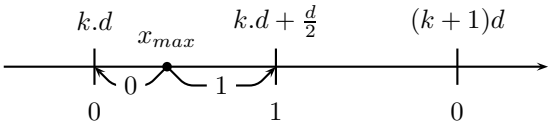


Figure 7.4: Encoding 0 or 1 into the main length x_{max} using quantization

The expansion coefficient of the polygon is defined as $\sigma = {x'}_{max}/x_{max}$. We transform each point $p = x.\vec{u} + y.\vec{v}$ in the original polygon into a point $p' = \sigma.x.\vec{u} + y.\vec{v}$ in the watermarked polygon.

$$p = x.\vec{u} + y.\vec{v} \quad \text{original point } p.$$
$$p' = \sigma.x.\vec{u} + y.\vec{v} \quad \text{modified point } p'.$$

The expansion is such that the maximum distortion on a vertex of a polygon is at most $d/2$. Remark that this distortion can be reached only for the vertices that are the farthest from the centroid along $\vec{u}$. On the average, and for these points, the actual distortion is $d/4$.

7.3.5 Watermarking Algorithm

Watermarking The complete algorithm is presented in Alg. 9. Let $1/\gamma$ be the target ratio of watermarked polygons. It is a parameter of the algorithm. For each polygon of the data set, we compute its robust identifier id. Then, we seed a pseudo-random number generator G (PRNG) with $\mathcal{K}.id$. If the first integer produced by G modulo γ is 0, we embed a bit in the polygon. The bit is chosen according to the next binary value produced by G and embedded using the previously described expansion method.

Variable Step Quantization We do not use a single quantization step d but a quantization interval $[d_{min}, d_{max}]$. Indeed, if d is the same for the whole data set, main lengths of all watermarked polygons will be multiples of d. This could be easily detected and used by an attacker to alter the watermark. To prove this, we measured the cumulative distributions of main lengths in three data sets: an unwatermarked one, a watermarked one using a fixed quantization step and another watermarked one using a variable quantization step, randomly chosen. For the two watermarked data sets, all polygons were watermarked ($\gamma = 1$) to emphasize differences. Graphs are shown on Fig. 7.5. We notice that, when the quantization step is fixed, the cumulative distribution is piecewise constant whereas it is very close to the original one when a variable quantization step is used. Fixed-step quantization is too visible through statistical analysis. So, we use a different quantization step for each watermarked polygon.

Algorithm 9: Watermarking algorithm: given a secret key $\mathcal{K}$, watermarking ratio $1/\gamma$, h, quantization step interval $D = [d_{min}, d_{max}]$, original data set (R_0, DB), outputs the watermarked dataset $(R_0, DB_{\mathcal{K}})$

```
foreach building P in DB do
    O ← Centroid(P)
    id ← hsb(O, h)                                   // robust identifier id
    seed(G, K · id)                                  // seed the PRNG G with K · id
    if NextInteger(G) mod γ = 0 then
                                                     // Watermark this building
        u⃗ ← orientation(P)                           // orientation
        x_max ← max{p ∈ P | Op⃗ · u⃗}                  // main length
        d ← d_min + NextFloat(G).(d_max − d_min)     // quantization step
        b ← NextInteger(G) mod 2                     // watermark bit b
        x'_max ← quantize(x_max, d, b)               // quantize x_max
        σ ← x'_max / x_max                           // expansion ratio
        Shrink(P, O, u⃗, σ)
        if collisions() then
            rollback()
```

Discussion Using this method, the watermark is spread almost uniformly over the data set. This process being controlled by a secret key, it is impossible to find the exact locations of expanded polygons, assuming the PRNG is secure. To alter the watermark, an attacker has to alter much more polygons than the watermarking process did if he wants to be sure to affect all watermarked polygons. The choice of watermarking parameters γ, d_{min} and d_{max} depends on the specific usage of the data set. They cannot be fixed arbitrarily for all applications but the following rules are always valid:

- there is an unavoidable trade-off between quality ($\gamma \uparrow, d_{min} \downarrow, d_{max} \downarrow$) and robustness of the watermark ($\gamma \downarrow, d_{min} \uparrow, d_{max} \uparrow$). Experiments presented in Section 7.4 give indications on how to choose optimal values;
- if the accuracy (maximum distance between a point in the data set and in the real world) of the unwatermarked database is β_1, then the accuracy of the watermarked one is $\beta_1 + d_{max}/2$. If the watermarked data set is sold under the agreement of an accuracy β_2, then d_{max} must be chosen so that $d_{max} < 2(\beta_2 - \beta_1)$;
- the allowed alteration on the building, i.e. d_{max} must be higher than the typing accuracy of the data set. Below this value, alterations can be considered as noise and rounded by a malicious user without altering the quality of the data set at all. For instance, a 1 millimeter alteration is meaningless in a data set of accuracy 1 meter.

7.3.6 Handling Data Constraints

The bit embedding method using expansion does not take into account topological relationships between buildings. We voluntarily chose to ignore them during bit embedding and to detect errors and cancel modifications when needed (function `collisions`). Such a strategy is valid as soon as few errors occur. By choosing $d_{max} = 4$ meters, the alteration on each point of a polygon is at most 2 meters. Usually, even in urban areas, polygons are spaced by a distance superior to 2 meters. Indeed, with this value, we got only one case of overlapping, even in the worst setting, i.e. when $\gamma = 1$. This validates the detect-and-cancel strategy. Such a post-watermarking filtering enables to handle any kind of errors which can occur sparsely during the watermarking process. Observe that since bit embedding is a local operation, collisions can only occur is a small area around the polygon. This allows the use of classical spatial indexing techniques to check for collisions.

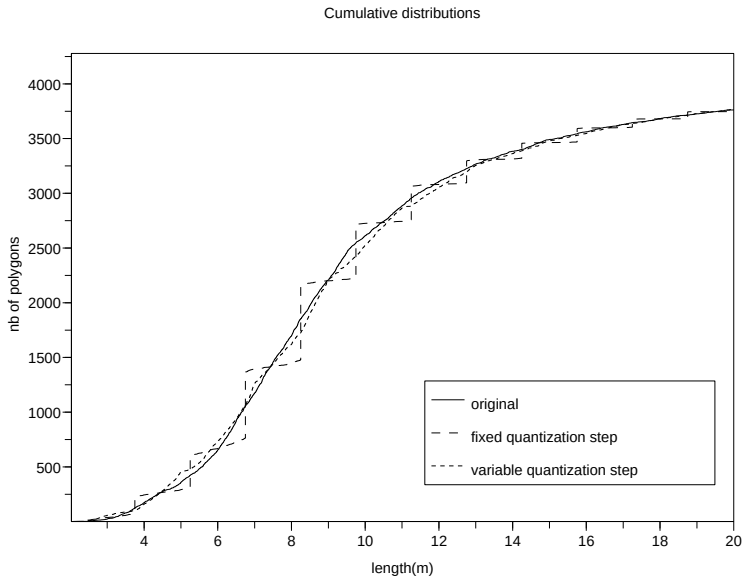


Figure 7.5: Statistical invisibility of watermarked data using variable step quantization

7.3.7 Detection

Outline Given a suspect data set (R', DB'), we translate it into the original reference system R_0. Then, we perform the actual detection which is very similar to the watermarking algorithm, with the essential difference that no alteration is performed. It consists of two steps: computing the ratio of matching polygons and comparing this ratio to a predefined threshold value α. The values of $d_{min}, d_{max}, h, \gamma$ and $\mathcal{K}$ used for detection must be the same as the ones used for watermarking. So they must be kept as part of the secret. For each of the polygon we seed a random generator with $\mathcal{K}$ concatenated with its identifier. If the polygon satisfies the watermarking condition (i.e. $\texttt{nextInteger}(G) \mod \gamma = 0$), we compute the expected bit value b as $\texttt{nextInteger}(G) \mod 2$. We also compute the quantization step d between d_{min} and d_{max}. Then, we decode the bit b' embedded in the main length x_{max} of the polygon and compare it with b.

Decoding To decode a bit from a quantized value x, we simply check whether it is one of the 1-quantizers or one of the 0-quantizers. If x is none of the i-quantizers, we compute the closest quantized value x'_1 in 1-quantizers and the closest quantized value x'_0 in 0-quantizers. We compare the distance $d_0 = |x'_0 - x|$ and $d_1 = |x'_1 - x|$. If $d_0 < d_1$, we decode a bit 0; if $d_0 > d_1$, we decode a bit 1. If $d_0 = d_1$ no bit can be decoded. Note that a quantized value, with step d, can be altered up to $d/4$ without leading to a decoding error. Quantization has been chosen because it optimizes the trade-off between average distortion (here, $d/2$) and the minimum alteration leading to a decoding error (here, $d/4$).

If the expected bit b and the decoded bit b' are the same, we say that the polygon matches. We maintain two counters, t (total) and m (match). The first one is incremented each time a polygon satisfying the watermarking condition is found. The second one is incremented each time this polygon carries the expected mark bit. Hence, the detection ratio m/t is the ratio of matching polygons.

It is easy to see that on a third party data set, the probability that each polygon matches is $1/2$. Therefore, the ratio m/t is compared to its expected value $1/2$ to decide whether the mark of the owner is present in the document or not. Practically, a detection threshold α must be set to bound the detection area. We detect a mark when $|m/t - 1/2| \geq \alpha$. The relevance of the detection process highly relies on the value of α. Prop. 64 gives the detection threshold for a maximum false positive occurrence probability f_p.

Proposition 64 *(direct application of [50]) Let p the number of polygons satisfying the watermarking conditions. If each polygon has a probability $1/2$ to match, the probability $\mathcal{P} = Pr\,(m/t - 1/2 \geq \alpha)$ is such that*

Algorithm 10: Detection algorithm: given a secret key $\mathcal{K}$, watermarking ratio $1/\gamma$, h, quantization step interval $D = [d_{min}, d_{max}]$, max. false positive occurrence probability f_p, reference system R_0 and (R', DB'), a suspect data set, outputs MARK or NO_MARK

```
1  Convert DB' into R_0
2  forall building P in DB do
3      O ← Centroid(P)
4      id ← hsb(O, h)                                   // robust identifier id
5      seed(G, K · id)                                  // seed the PRNG G with K · id
6      if NextInteger(G) mod γ = 0 then
7          t++                                          // increment total count
8          u⃗ ← orientation(P)                           // orientation
9          x_max ← max{p ∈ P | Op⃗ · u⃗}                  // main length
10         d ← d_min + NextFloat(G).(d_max − d_min)
11         b ← NextInteger(G) mod 2                     // expected bit b
12         x'_0 ← quantize(x_max, d, 0)                 // closest 0-quantizer
13         x'_1 ← quantize(x_max, d, 1)                 // closest 1-quantizer
14         if |x_max − x'_0| > |x_max − x'_1| then
15             b' ← 1                                   // found bit is b' = 1
16         else
17             b' ← 0                                   // found bit is b' = 0
18         if b = b' then
19             m++                                      // increment match count
20 α ← Threshold(f_p, t)
21 if |m/t − 1/2| > α then
22     Return MARK
23 else
24     Return NO_MARK
```

$\mathcal{P} \leq e^{-2\alpha t^2}$.

Then, the false positive occurrence probability defined as $f = \mathcal{P}(|m/t - 1/2| \geq \alpha)$ is such that $f \leq 2e^{-2\alpha t^2}$. Choosing $\alpha = -\log(\delta/2)/2t$ as the detection threshold permits to keep f under δ. We use this formula in our experiments to keep false positives occurrence probability under $\delta_0 = 10^{-4}$.

7.4 Experiments

7.4.1 Framework

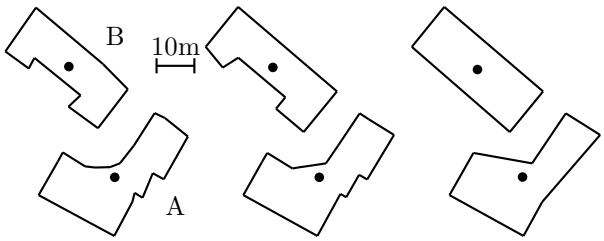
Data All experiments presented in this section (except speed ones) were realized on buildings from the French city of Pamiers. The data is part of the **BD TOPO™** [54], a topological database product from the French National Mapping Agency (IGN), the major maps provider on the French market. The product consists of several coherent layers (hydrographic network, roads, buildings...) from which we extracted only the buildings layer. This layer is composed of 4 278 polygons (35 565 vertices), representing dense build areas (downtown – west side) as well as sparse ones (residential blocks – east side). It has a contractual accuracy of 1 meter. A glimpse on the Pamiers data set is given in Fig. 7.6.

Software We developed a Java version of our algorithm and packaged it as a library for our generic database watermarking framework [67]. The data set was stored in a Postgresql/Postgis database. Advanced geographic functions from GeOxygene [23] were also used. GeOxygene is an open source application

Figure 7.6: Pamiers buildings (extract)

developed at IGN. It aims at providing an open framework which implements OGC/ISO specifications for the development and deployment of geographic applications. The core watermarking method is available under the GPL license [64].

Filters/Attacks We performed an extensive series of experiments to validate the robustness of our method. All the filters/attacks presented in Section 7.2.3 were tested.



(a) Original (b) AKH (c) DP($d = 5m$)

Figure 7.7: Examples of filters

Protocol We consider that an attack is successful if it destroys the watermark with high probability while inducing a quality loss comparable to the one introduced by the watermarking process. In a same manner, we consider that a watermarking algorithm A is better than an algorithm B against a specific attack if it is as robust as B while inducing a quality loss significantly smaller than B. To emphasize the benefits of

watermarking by expansion, we put side by side a random noise based method and ours and compare them in terms of robustness/distortion trade-offs. So, we begin by quantifying the quality losses of both schemes.

7.4.2 Impact of Watermarking on Quality

To evaluate the impact of our watermarking algorithm on the Pamiers data set, we applied it with different watermarking ratios $1/\gamma$, with $\gamma \in \{10, 15, \ldots, 100\}$, and different quantization ranges $[d_{min}, d_{max}] \in \{[1,2],[3,4],[5,6]\}$. These ranges start from data accuracy (1 meter) to the maximum reasonable alteration (6 meters).

Average Accuracy Alteration The impact on the accuracy is displayed on Fig. 7.8(a). Alteration increases when quantization steps increase and when γ decreases. We observe that the average alteration of accuracy is proportional to $(d_{min} + d_{max})/\gamma$. For instance, when $[d_{min}, d_{max}] = [3,4]$, a good approximation of the average alteration of accuracy is $0.07.(d_{min} + d_{max})/\gamma$. The ratio $1/\gamma$ is not surprising since on the average, $1/\gamma$ polygons are watermarked. Furthermore, the expected alteration for the farthest vertex from the centroid is $(d_{min}+d_{max})/4$. The alteration of all the points from a watermarked polygon are proportional to the alteration of this particular vertex. These dependencies can be used by the watermarker to choose the parameters of the marking algorithm: if d_{max} is obtained as the maximum allowed alteration, and if the target alteration is fixed, one can easily compute γ.

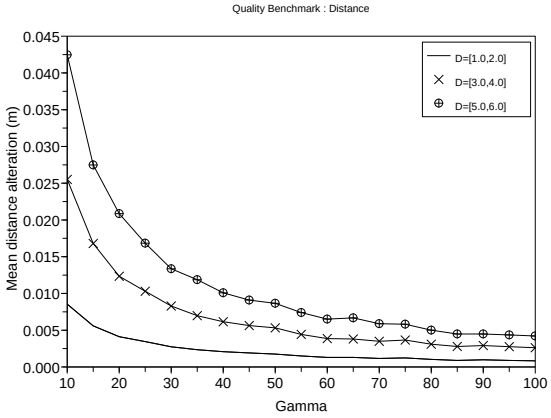
Maximum Accuracy Alteration The behavior is similar to the one obtained for average accuracy alteration.

Angular Quality We verify that the variation of angular quality introduced by our method is negligible on Fig. 7.8(b). Even for the highest quantization steps, $[d_{min}, d_{max}] = [5,6]$, the highest angular energy variation is at most $+0.08$. As a comparison, a weak gaussian noise (deviation $d = 0.2m$) increases the angular energy by $+6.19$.

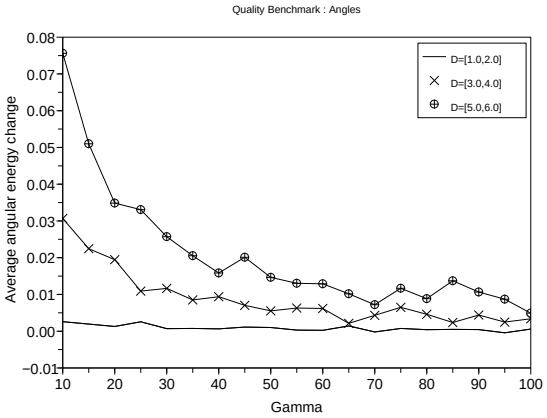
Area Modification The impact of the watermarking algorithm on the areas of the polygons of the data set is displayed on Fig. 7.9. On Fig. 7.9(a) is displayed the maximum relative area alteration. Its is proportional to the mean of minimum and maximum quantization steps. On Fig. 7.9(b) is displayed the average signed area alteration, which shows that alterations compensate themselves, a good property. On Fig. 7.9(c) is displayed the average relative area alteration. Its behavior is similar to average accuracy alteration.

Detection: False Positives On Fig. 7.10 are displayed the detection ratios obtained with different secret keys on a previously watermarked data set. A hundred different keys were tested, among which the one used for watermarking. Detection threshold $\alpha \approx 15\%$ was chosen so that false positive occurrence probability is at most 10^{-4}. It appears that only the secret key $\mathcal{K} = 100$ used in the embedding process leads to a positive detection of mark. This experimentally validates the relevance of the detection algorithm.

Watermarking Speed To evaluate the watermarking rate our implementation can achieve, we used a significantly larger data set, still extracted from the **BD TOPO™**. It consists of the building layer from the Pyrénées Orientales (64), a French department. It contains 243 201 polygons (1 880 405 points and 65 megabytes for the geometry). Using $\gamma = 20$, the whole data set was watermarked in 74 mins 30s (including database I/O times),i.e. more than 50 polygons per second. The detection process took 1min 38s. A simple notebook powered by an Intel Core Duo processor running at 1.6Ghz with 2Gb of RAM and a 7200rpm hard drive was used for this experiment.



(a) Mean accuracy



(b) Angular energy

Figure 7.8: Impact of our watermarking method on quality

7.4.3 Comparison with Random-Noise Based Watermarking

The random noise scheme is based on the insertion of a pseudo-random noise within the data. Even if it is not a scheme found in the literature, it supersedes existing ones and mimic the behavior of others. It is similar, in most cases, to the schemes that introduce perturbation in the data set without taking into account the shapes of the polygons [83]. The point is that if shapes are expected to be regular, a watermarking algorithm working by altering these shapes is suboptimal in terms of robustness/distortion trade-off. Indeed, most users will correct the shapes so that the regularity aspect is brought back. This operation is very likely to remove the watermark.

The algorithm consists of looping over all the vertices of the data set. The x and y coordinates of the vertices are watermarked independently. For each coordinate, a PRNG is seeded with its highest significant bits. Only the least significant bits are altered. Their positions and values are determined according to the drawings of the PRNG. This method can be seen as a straightforward extension of the existing AKH [7] scheme in which most significant bits of the coordinates are used as primary keys and least significant bits as alterable attributes.

The quality loss introduced by such a random noise scheme is controlled by three parameters: the watermarking ratio $1/\gamma$, the least highest significant bit $lspow2$ and the number $\xi = 2$ of alterable powers of two. For instance, if $lspow2 = 2$ and $\xi = 2$, the maximum distortion on a coordinate is $2^{lspow2-1} = 2$ and the minimum distortion is $2^{lspow2-\xi} = 1$. Using higher values of ξ enables for extra embedding bandwidth

but lead to distortions that are removed by any rounding of the coordinates. In what follows, we compare our scheme (with $d_{min} = 3$ and $d_{max} = 4$) with two random noise schemes parameterized with two sets of parameters. For the first one, RNW1, $lspow2 = -1, \xi = 1$; for the second one, RNW2, $lspow2 = 2, \xi = 2$. These values were chosen for the following reasons: RNW1 achieves an average accuracy alteration (compare Fig. 7.8(a) and Fig. 7.11(a) - both curves decrease as $1/\gamma$ and have a mean accuracy alteration of 0.025 for $\gamma = 10$) very close to the one observed using our method) whereas RWN2 has a maximum alteration on each vertex equals to ours.

7.4.4 Robustness against Filters and Attacks

For all the experiments, we present the detection ratios observed after application of a filter, together with the detection threshold of the mark (computed for a false positive occurrence probability of 10^{-4}). We put side by side the results obtained using our method, RNW1, RNW2 and the combined scheme presented in Section 7.4.5.

Squaring We present on Fig. 7.13, the detection ratios observed after squaring. Three different values of the maximum allowed point position alteration were tested, namely $d = 1.0$ (commonly used value), $d = 1.5$ and $d = 2.0$ (very aggressive squaring, used here only for validation). For our scheme, squaring has no effect on the detection ratio no matter the watermarking parameters. Even in a worst case scenario, $\gamma = 100$ and $d_{max} = 2$, the watermark is still detected. For RNW1 and RWN2, the detection ratio is considerably lowered when the data is squared.

Douglas & Peucker Simplification The Douglas-Peucker algorithm [29] is a polyline simplification algorithm. It is systematically used by geographic data users with a small factor to filter points of the database. It consists of pruning the points of a polyline whose distance to the line joining the polyline bounds is too small. The distance threshold under which these points are pruned is called d. The higher this threshold, the more aggressive the algorithm is. An example of Douglas-Peucker simplification with $d = 5$ meters is given on Fig. 7.7(c). We tested the robustness of our algorithm against Douglas-Peucker filtering for different threshold values. The results are displayed on Fig. 7.14. Note that our algorithm behaves very well when the Douglas-Peucker filtering distance is 1 or 2 meters. The detection ratio never falls below the detection threshold. The reason is that the shapes of the polygons are regular enough so that they are invariant to these filters. For a filtering distance of 5 meters, the mark is removed in some situations. But such a filtering is very aggressive: it can remove a $4m \times 4m$ room in a building.

Cropping Observe that the whole data set is not necessarily interesting for a malicious user, since profit can still be made from the (illegal) redistribution of a subset. Our method does not require the whole suspect data set to perform detection. This is illustrated on Fig. 7.15 for our scheme. We randomly generated rectangular subsets of the watermarked data set and performed detection on them. On the x-axis is displayed the number of polygons in a crop while on the y-axis we plotted the number of watermarked and matching polygons against the number of polygons in a crop. We also added the minimum number of matching polygons that must be found to detect the watermark, i.e. the detection threshold. In all cases the watermark is detected. Furthermore, the curves are, once removed local artifacts, almost linear. This indicates that the distribution of the watermarked polygons over the data set is nearly uniform. For other schemes, experiments not presented here show that the repartition of watermarked polygons is nearly uniform so detection is not too much affected by squaring.

Gaussian Noise Gaussian noises with deviations of $20cm$, $60cm$ and $1m$ were tested and the results displayed in Fig. 7.16. For reasonable noises ($d = 20cm$ or $d = 60cm$), marks are still detected. When $d = 1m$, the watermark is removed for a large range of γ. Interestingly, the application of a squaring algorithm on a noised data set increases the detection ratio for our scheme. It even permits the recovery of the watermark for higher values of γ. RNW1, RNW2 and the combined scheme show similar behaviors.

Applying a New Watermark On Fig. 7.17(a), we show how the detection algorithm copes with a new application of the watermarking algorithm with a different key. Different watermarking parameters for the attack were tested: highest quantization steps and lowest γ. For all parameters, the observed detection threshold is such that the first watermark is still detected. Applying another watermark has a limited effect on the first watermark. This is due to the fact that the two watermarks are embedded into distinct polygons. Moreover, on the sparse polygons chosen simultaneously by the two watermark embeddings, the second watermark is the only one to be detected on these polygons. RNW1, RNW2 and the combined scheme show similar behaviors.

Enlarge to rectangle filter For this experiment, we used the Map Generalization Toolbox of OpenJump [3]. The *enlarge to rectangle* filter consists of replacing each building of the max with its minimum bounding rectangle. If a map is to be drawn from the geodata, its scale can be passed as an argument to the enlarge to rectangle filter. In this case, illegible buildings are replaced by their minimum bounding rectangle and enlarged so that they are visible. Larger buildings are not modified. The threshold values are chosen according to the Swiss Society of Cartography [108], e.g. 0.25mm for width and $0.35mm \times 0.35mm$ for buildings minimum size on maps. The results of the experiments are presented on Fig. 7.18. They show that replacing all buildings by their bounding rectangle washes out the watermark. When a watermarked data set is used for map generation, the detection ratio decreases as the scale increases. This is not surprising since a high scale implies that many buildings are represented by a minimal area rectangle. So bits embedded in such buildings can not be recovered. Nevertheless, it is interesting to notice that watermarking one polygon out of ten (i.e. $\gamma = 10$) enables to recover marks in a map with scale 25000. Compared to RNW1 and RNW2, our scheme performs better.

Elongation Change Filter The elongation change filter consists of multiplying the main length of all polygons with a fixed factor. For experiments presented on Fig. 7.19, ratios 0.85,0.9,...,1.2 were used. Hopefully, when the ratio is 1, the detection ratio is close to 1. If the ratio is between 0.95 and 1.05, the watermark is still detected. This means that an attacker who wants to ensure that the watermark is removed must alter the main length of *all* polygons by more than 5%. Compared to RNW2 and RNW2, our scheme is not as good since elongation ratios 0.9 and 1.1 remove the watermark embedded using our scheme whereas the ones embedded using random noise based scheme are still detectable. Elongation change filter is the worse attack our scheme can encounter since it precisely affects the embedding area used for watermarking. But this attack results in a poor quality data set, whose accuracy is lowered for all polygons (see also Section 7.4.5).

Minimum Bounding Rectangle Filter This filter replaces each polygon by its minimum bounding rectangle. This filter is very aggressive, too for both algorithms since the watermark is removed for a large range of γ in all cases. But, as it can be seen on Fig. 7.20, our scheme performs slightly better.

Missing Reference System Without a reference system, a data set is useless for automatic operation and interoperability. Meanwhile, the suspect data set can be mapped into the original reference system to perform the detection process. This mapping operation is quite common and easy, as soon as (a reasonably large part of) the original data set is available.

Other Attacks Many other kinds of attacks may be envisioned. The mixing attack consists of mixing a portion of a watermarked data set with an unwatermarked one. This attack is expected to lower the detection ratio but not render the watermark unreadable.

The aliasing attack consists of switching the edges of the polygon with zig-zag shaped sequences of edges. The goal is to modify the orientation of the polygon. But it implies adding a huge number of extra (fake) points to the database and altering the overall shapes of polygons. Furthermore, the artifact needs to have an amplitude exceeding angle tolerance ratio.

Note also that our method is invariant to polygons rotation since the expansion coefficient is defined relatively to polygon orientation and not to a particular reference system.

We also observe that, for our scheme, squaring the data prior to the watermarking process, beside increasing its value, enhances slightly the detection ratio.

7.4.5 Discussion

Table 7.2 sums up robustness experiments. Robustness results for our method are presented in the WM column. Whether a method achieves robustness is indicated with the appropriate Yes/No status. When a method is robust as far as enough polygons are watermarked, the threshold watermarking ratio under which the method has been experimentally proven robust is also indicated. Remark that robustness might also be achieved for higher values of γ. In the second and third columns are shown the average quality loss introduced by the attacks. These must be put into balance with the distortion introduced by watermarking algorithms (see Section 7.4.2 and 7.4.3).

As we expected, our method resist all kinds of squaring, including the most destructive ones. On the contrary, watermarks embedded using random-noise based algorithms are washed out. Our method shows also robustness against the majority of attacks. The fact that GN(d=1m) and DP(d=5m) erase the watermark must be put into balance with the quality losses these attacks introduce. They tremendously reduce the quality of the data set. Furthermore, it behaves well against change elongation filters since the length of polygons must be increased or decreased by more than 10% to ensure watermark removal. But this remains the weak point of our method since the embedding area is directly affected.

Compared to random-based schemes, our algorithm performs better in the majority of cases. Whereas RWN1 and RWN2 are more robust against the change elongation filter, they have a significantly higher impact on quality compared to our method. In that sense, the experiments show that our algorithm achieves a very good distortion/robustness trade-off.

In the last column of the table, we combined our scheme and RNW2 into an *expand-and-translate* scheme. In a first time, we apply our scheme and then apply RNW2 on the centroid of the polygon (left invariant by the first step). We obtain a modification of the coordinates of the centroid which is used to translate the polygon. Obviously, this combined scheme introduces a larger quality loss but it achieves robustness for all the filters/attacks used in the present work. It can be preferred to our basic scheme if robustness is to supersede quality for a specific application.

7.5 Related Work

In our database approach, polygons are stored in a relational database management systems enriched with geographical features. Since polygons are stored in relational tables, state-of-the art watermarking algorithms for relational databases might have been used. It happens that least significant bits modifications used in previous works [7] can not be mapped onto our geographical setting. It is easy to alter least significant bits of points of the map but the angular quality is not taken into account (on the contrary, least significant bits methods may perform well on simple points databases, like point-of-interest data sets in use in GPS viewers). Methods described in [39, 104] allow for the description of usability queries to be preserved by watermarking. But they either focus on basic numerical aggregates like `SUM` queries [39], which are not rich enough to represent angular constraints, or based on a trial and error method to handle generic black-box queries [104]. Using the latter method, it might not be possible to reach a valid set of alterations since no search strategy is defined.

Despite that state-of-the art is very rich on watermarking still images which can be directly applied to image maps, fewer works were carried on on watermarking vector maps. A complete study of vector maps watermarking [83] has been recently published which divides this field into three categories: spatial domain watermarking, transform domain (DCT,DWT,...) watermarking and 3D meshes adapted algorithms. Spatial domain watermarking consists of modifying directly the coordinates of the vertices of the data set. Patch-based techniques [56, 84, 93, 115] are based on a decomposition of the data set into patches in which

Table 7.2: Robustness and impact of watermarking

Filter	Precision alt.	Angular energy	WM	RNW1	RNW2	Combined
SQ (d=1m)	0.19	-14.1	**Yes**	N°	**Yes**	**Yes**
SQ (d=1.5m)	0.23	-16.2	**Yes**	N°	$\gamma < 90$	**Yes**
SQ (d=2m)	0.27	-17.2	**Yes**	N°	$\gamma < 80$	**Yes**
DP (d=1m)	N/A	- 1.3	**Yes**	**Yes**	**Yes**	**Yes**
DP (d=2m)	N/A	- 0.3	**Yes**	**Yes**	**Yes**	**Yes**
DP (d=5m)	N/A	68.1	$\gamma < 70$	**Yes**	**Yes**	$\gamma < 100$
CA	0	0	**Yes**	**Yes**	**Yes**	**Yes**
GN (d=0.2m)	0.18	6.19	**Yes**	$\gamma < 30$	**Yes**	**Yes**
GN (d=0.6m)	0.53	40.00	**Yes**	N°	$\gamma < 75$	**Yes**
GN (d=1m)	0.89	78.52	$\gamma < 65$	N°	$\gamma < 30$	$\gamma < 75$
OW	-	-	**Yes**	**Yes**	**Yes**	**Yes**
ETR	N/A	-6.53	$\gamma < 40$	N°	$\gamma < 25$	$\gamma < 50$
ETR(scale=25000)	N/A	-6.06	N°	N°	N°	$\gamma < 40$
ETR(scale=250000)	N/A	-0.03	**Yes**	**Yes**	**Yes**	**Yes**
CE(scale=0.90)	1.06	0.16	N°	N°	$\gamma < 45$	**Yes**
CE(scale=0.95)	0.53	0.02	$\gamma < 95$	N°	**Yes**	**Yes**
CE(scale=1.0)	0	0	**Yes**	**Yes**	**Yes**	**Yes**
CE(scale=1.05)	0.53	0.09	$\gamma < 95$	**Yes**	**Yes**	**Yes**
CE(scale=1.10)	1.06	0.27	N°	N°	$\gamma < 50$	**Yes**
MBR	N/A	-6.6	$\gamma < 75$	$\gamma < 35$	$\gamma < 50$	$\gamma < 75$

bits are embedded by vertices translation [84] and/or data distribution within a patch [56, 93, 115]. Such schemes are sensitive to patch decomposition that can vary when the data set is cropped. Schemes of [56,93] create detectable nodes aggregations; [84] is not blind and [115] does not respect the shape of smooth objects (including squared buildings).

Vertices addition based [52, 75, 87] techniques are the best from a quality viewpoint. They consist of interpolating the existing edges of the data set with fake points. In the context of buildings where the shapes are regular, such interpolations are easily detected and removed by simplification algorithms. In our context, these schemes are not robust enough. The least-significant bit watermarking method presented in [114] preserves accuracy but is very sensitive to vertices addition, which destroys synchronization.

In [100], a high watermarking capacity algorithm for vector maps is introduced. It is based on a previous work on 3D meshes watermarking by the same authors. It is robust against common geographical filters like Douglas-Peucker simplification algorithm. It is based on a decomposition of the database into patches and by moving points of a common patch into a subpatch to embed the watermark. Nevertheless, efficient attacks erasing the watermark without destroying the quality of the data set can be easily planned, as the method requires known synchronization points. Another approach followed by [13] synchronizes the watermark based on a triangular mesh decomposition. Within secretely chosen groups of adjacent triangles, a vertex is moved in order to hide a bit of information.

Our method shares a common skeleton with the popular AHK algorithm [7]. But it differs on several aspects: it does not require primary keys, and do not use high significant bits to replace them in a straightforward manner (instead the centroid is used); the bit embedding operation is not a least significant bit embedding, which is very fragile against rounding, but a quantization method.

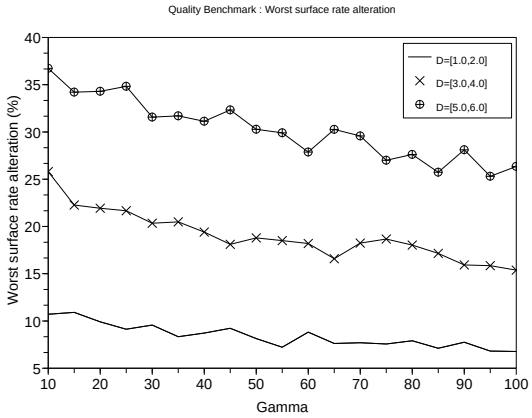
7.6 Conclusion

In this section we presented a blind watermarking algorithm for polygon data sets. It is well suited to building layers of geographical data sets since watermarks are invariant through aggressive geographical filters applied by data users, including squaring and replacement by bounding boxes. We experimentally showed that it is difficult for an attacker to erase the watermark without paying an extra quality fee, compared to watermarking. The algorithm has been implemented into an open database watermarking framework [67] and is available online [64]. We are currently working on designing algorithms for other layers of geographical

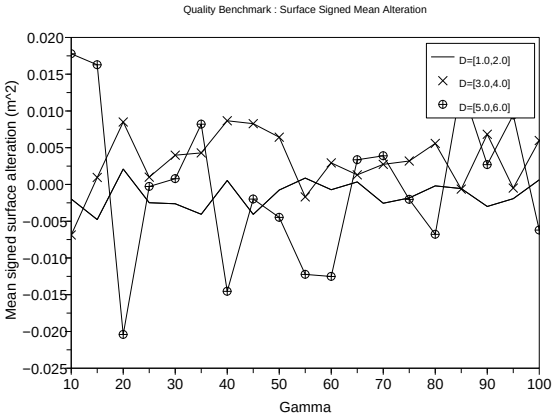
data sets. A real challenge we are faced with is to deal with the interactions between the different layers. Indeed, watermarking algorithms must be adapted to the data; there is no unique solution. Even if we know how to perform watermarking and detection on a single layer, it is challenging to orchestrate several algorithms on several layers so that resulting watermarked layers remain consistent.

Related publications

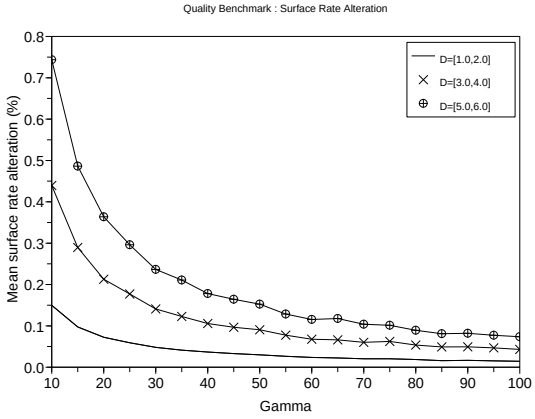
- Julien Lafaye, Jean Béguec, David Gross-Amblard and Anne Ruas. Invisible Graffiti on your Buildings: Blind & Squaring-proof Watermarking of Geographical Databases. In *10th International Symposium on Spatial and Temporal Databases (SSTD)*, July 16-18, 2007, Boston. LNCS 4605, pages 312-329.



(a) Maximum relative area alteration



(b) Average signed area alteration



(c) Average relative area alteration

Figure 7.9: Impact of our watermarking method on areas

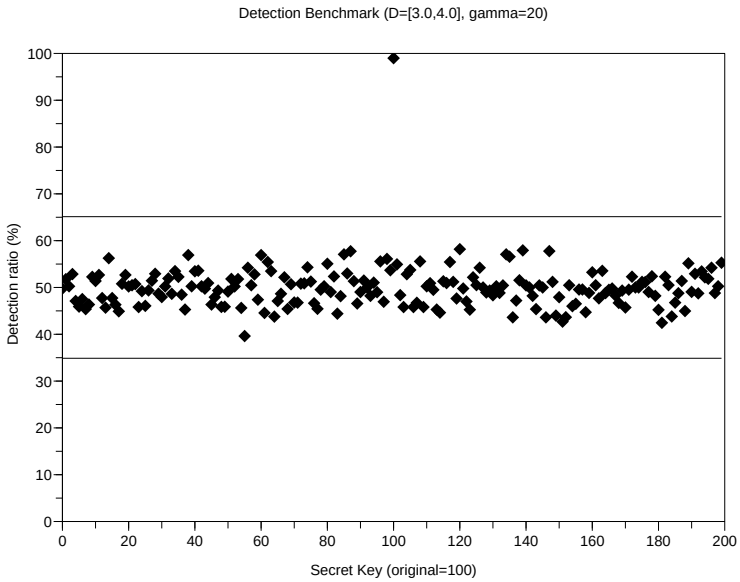
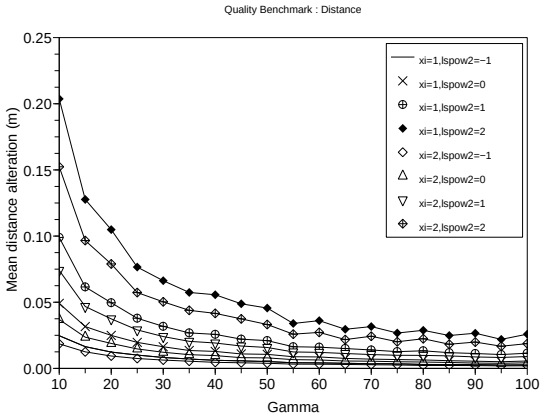
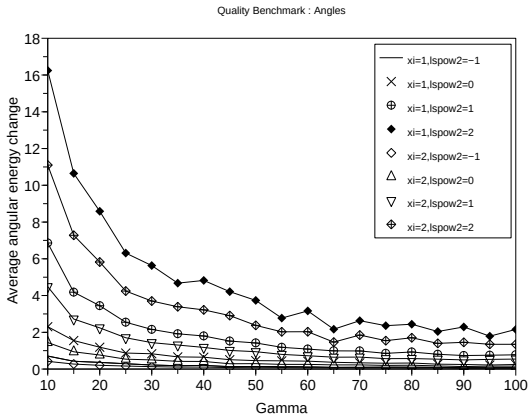


Figure 7.10: Undetectability of the mark without the correct key

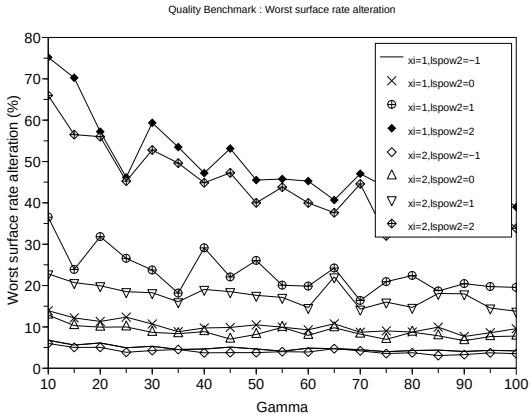


(a) Mean accuracy

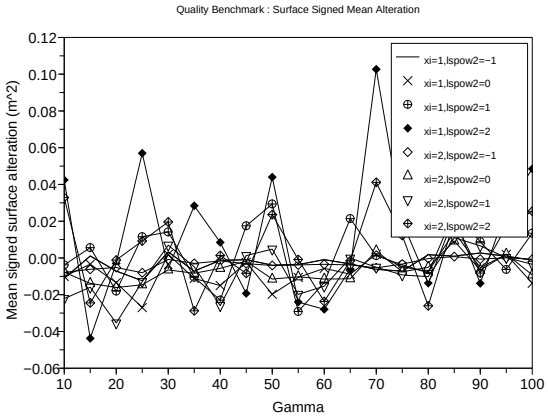


(b) Angular energy

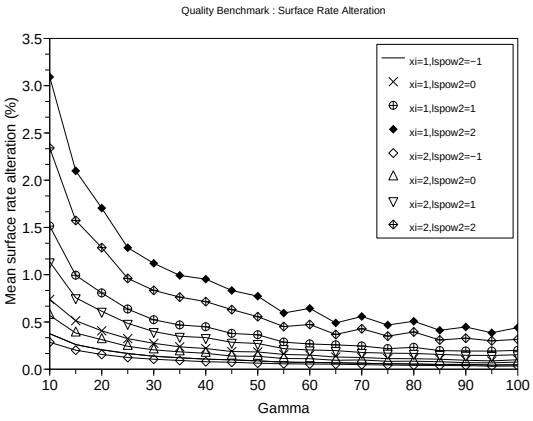
Figure 7.11: Impact of random noise watermarking on quality



(a) Maximum relative area alteration

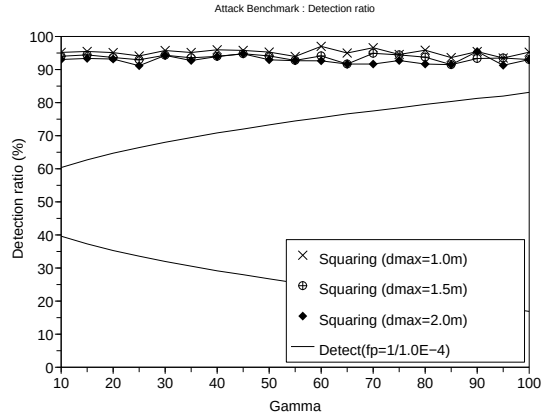


(b) Average signed area alteration

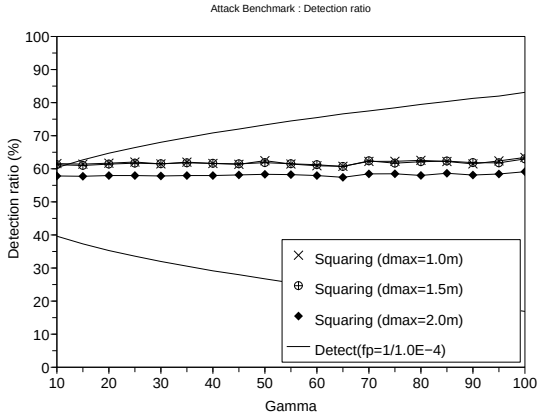


(c) Average relative area alteration

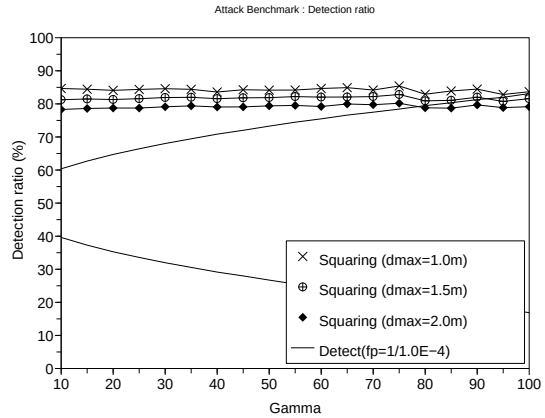
Figure 7.12: Impact of random noise watermarking on areas



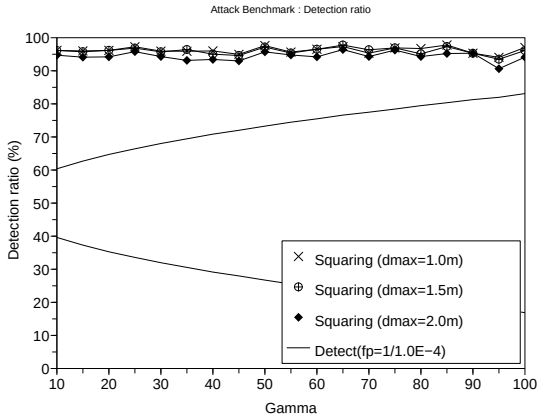
(a) WM



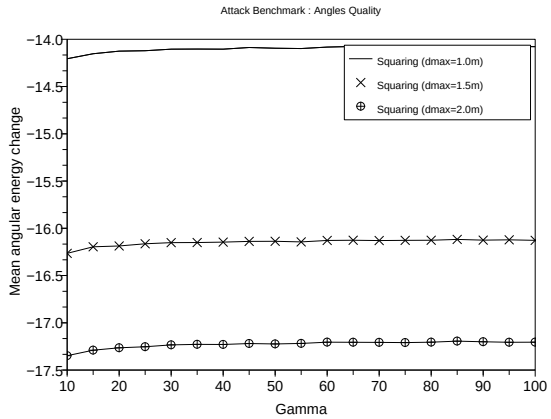
(b) RNW1



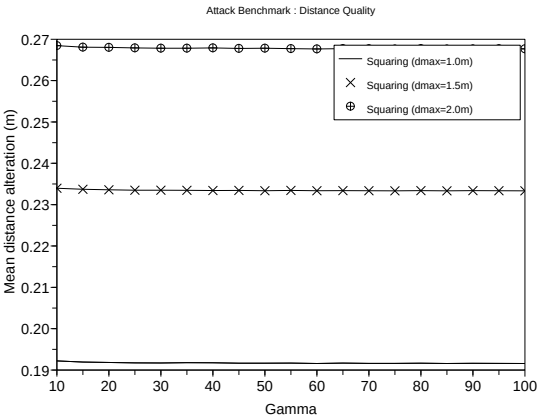
(c) RWN2



(d) Combined

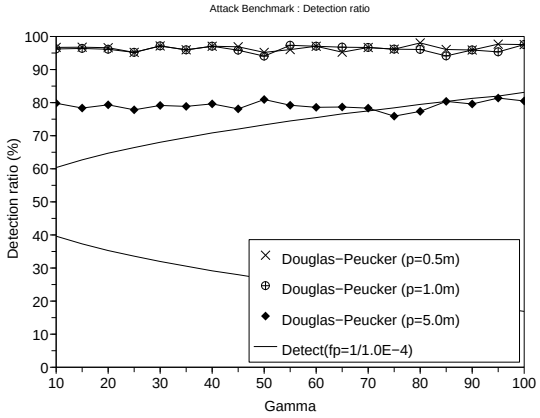


(e) Angular quality

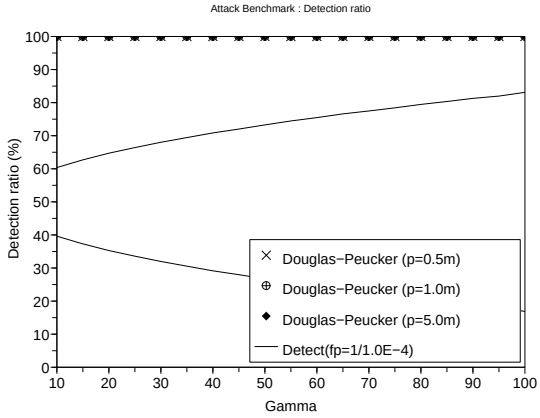


(f) Accuracy alteration

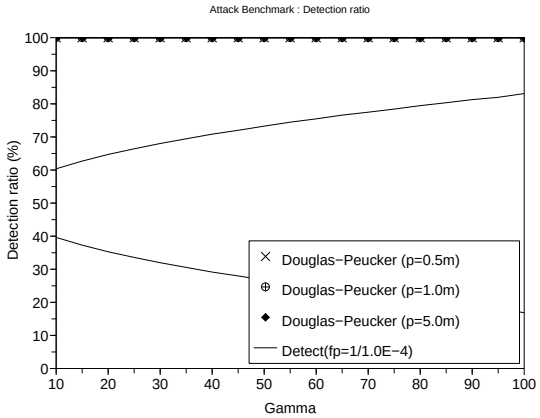
Figure 7.13: Squaring filter



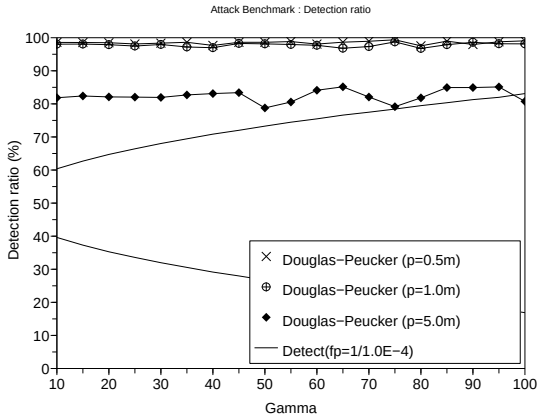
(a) WM



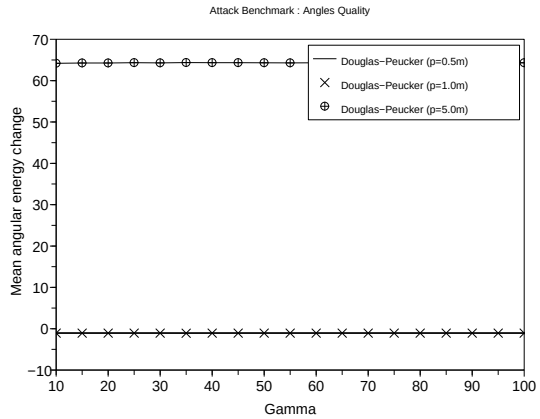
(b) RNW1



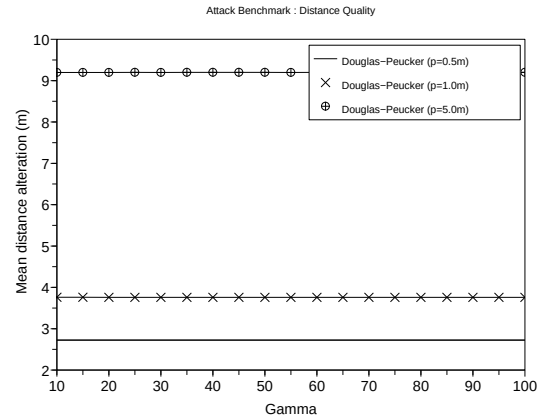
(c) RWN2



(d) Combined



(e) Angular quality



(f) Accuracy alteration

Figure 7.14: Douglas-Peucker simplification

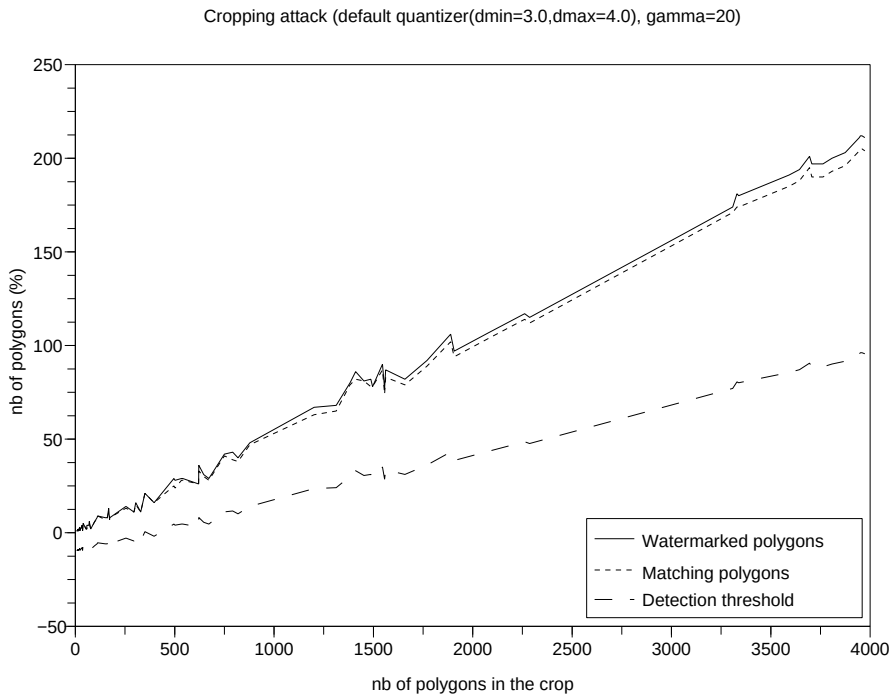
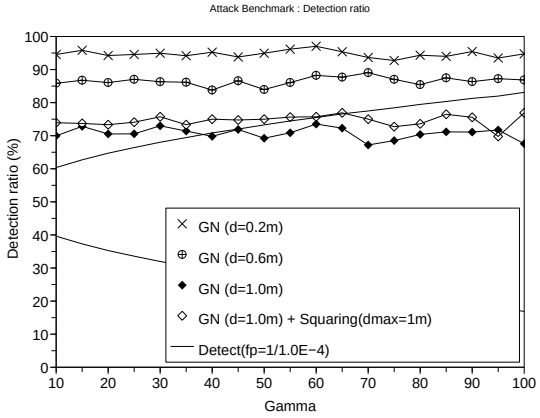
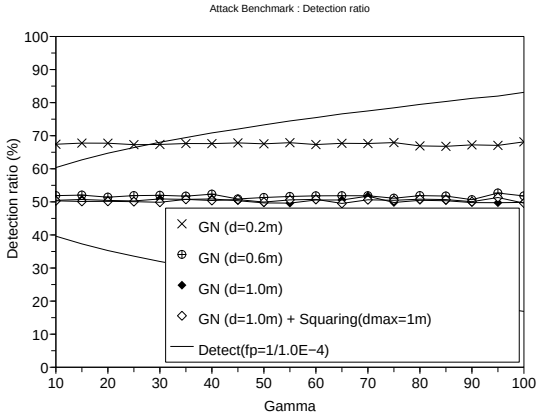


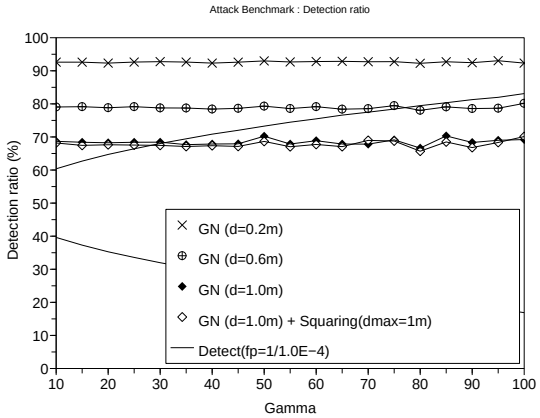
Figure 7.15: Cropping attack



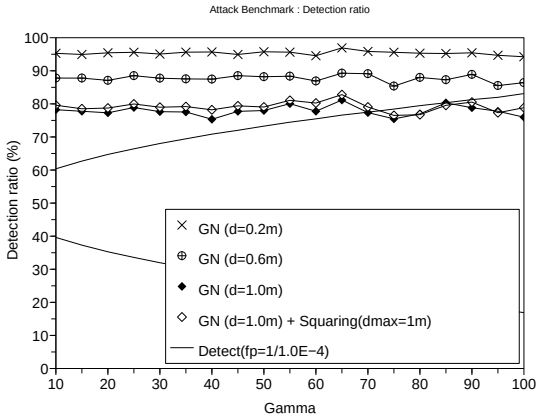
(a) WM



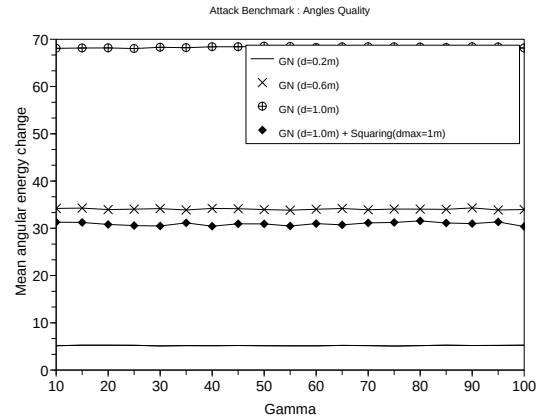
(b) RNW1



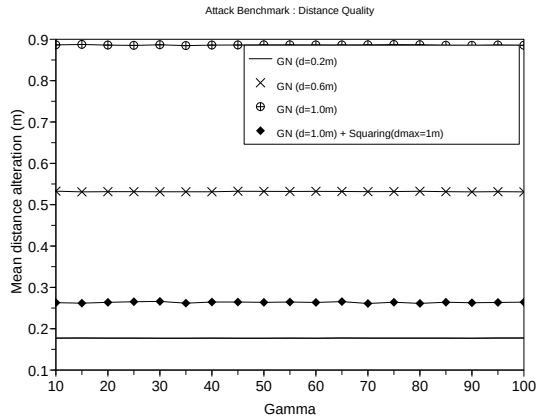
(c) RWN2



(d) Combined

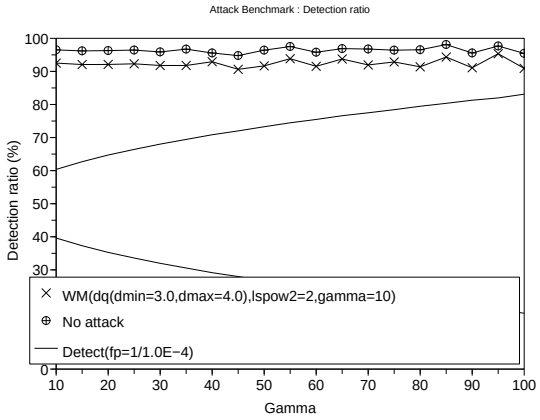


(e) Angular quality

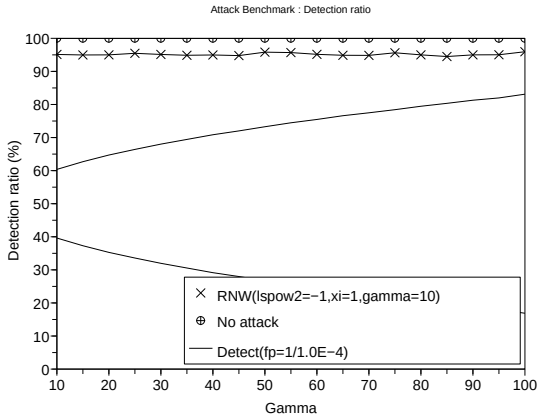


(f) Accuracy alteration

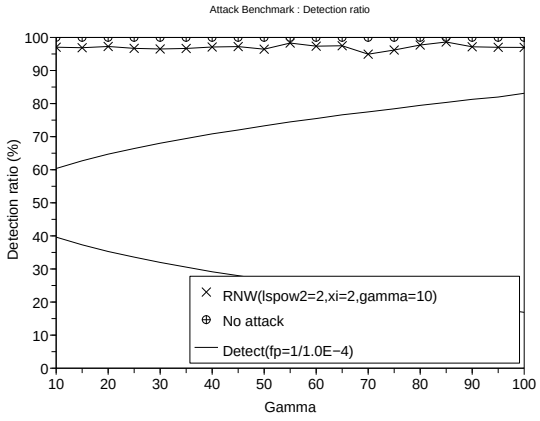
Figure 7.16: Gaussian noise



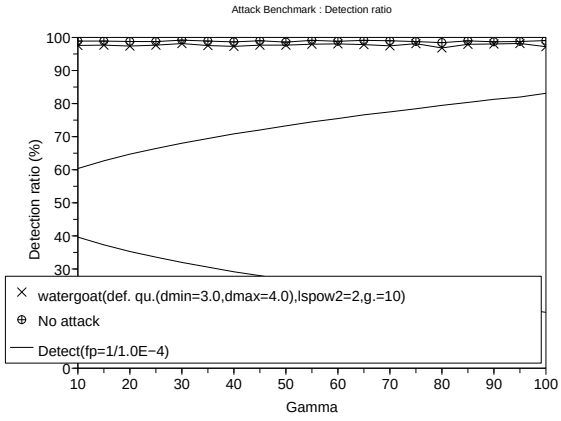
(a) WM



(b) RNW1

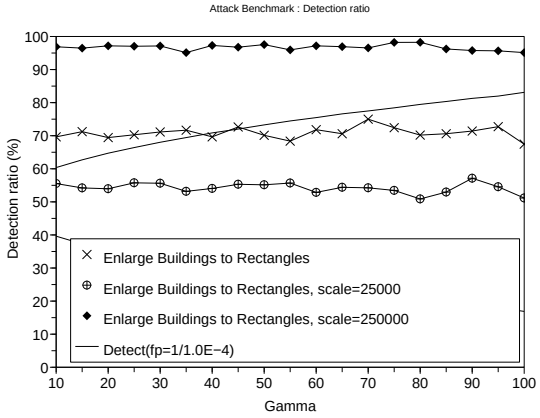


(c) RWN2

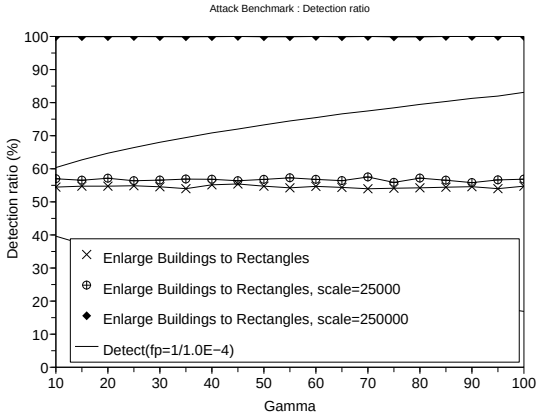


(d) Combined

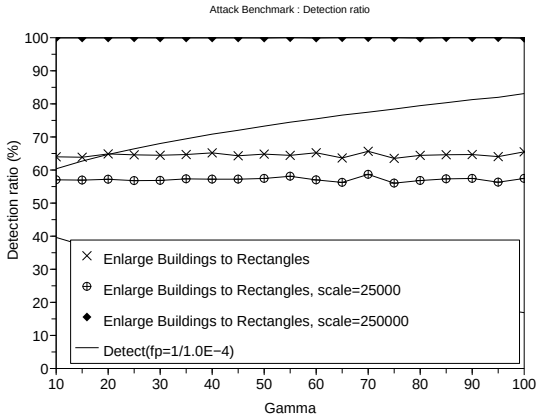
Figure 7.17: Over-watermarking



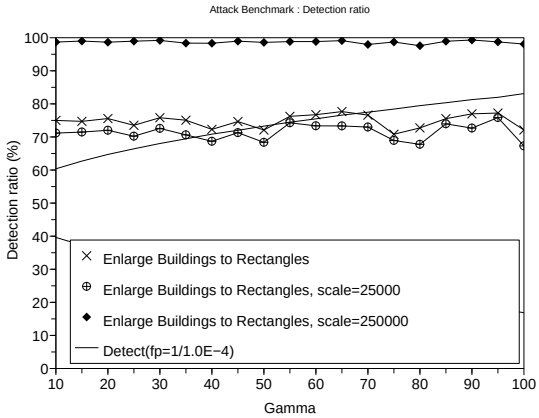
(a) WM



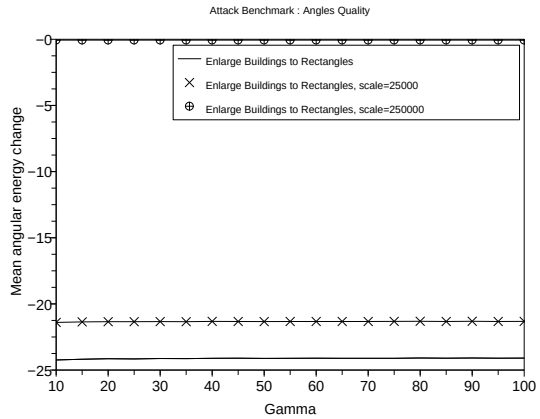
(b) RNW1



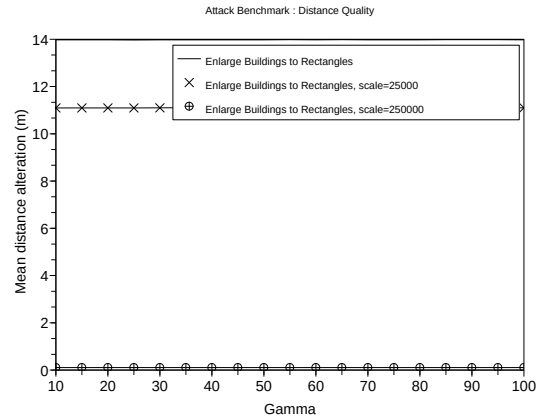
(c) RWN2



(d) Combined

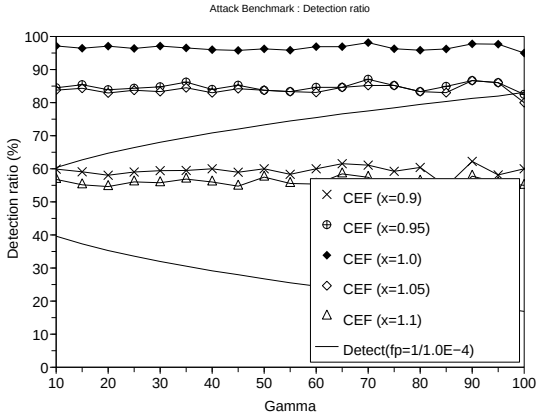


(e) Angular quality

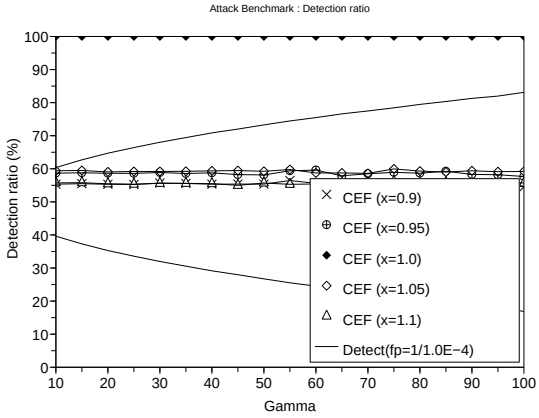


(f) Accuracy alteration

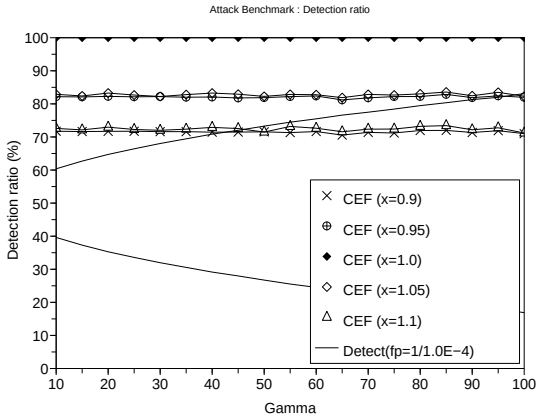
Figure 7.18: Enlarge to rectangle filters



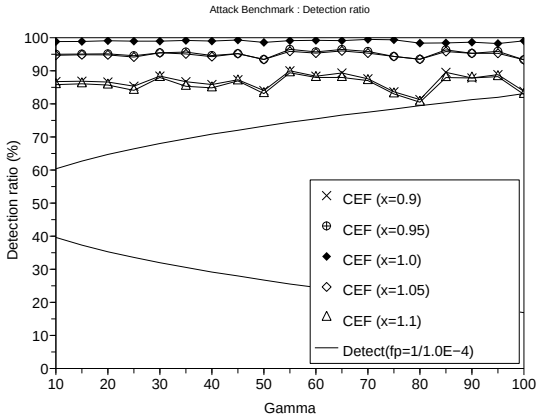
(a) WM



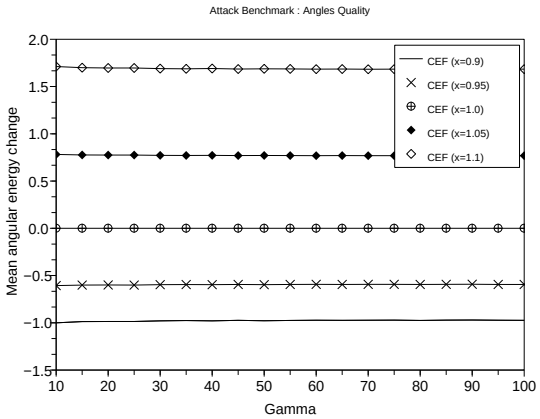
(b) RNW1



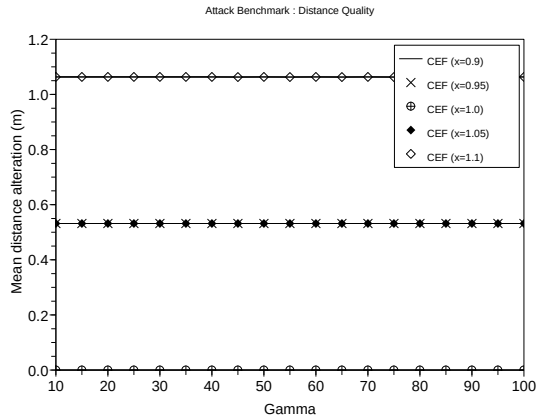
(c) RWN2



(d) Combined

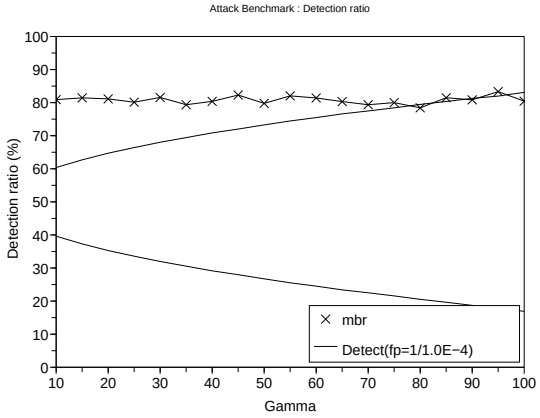


(e) Angular quality

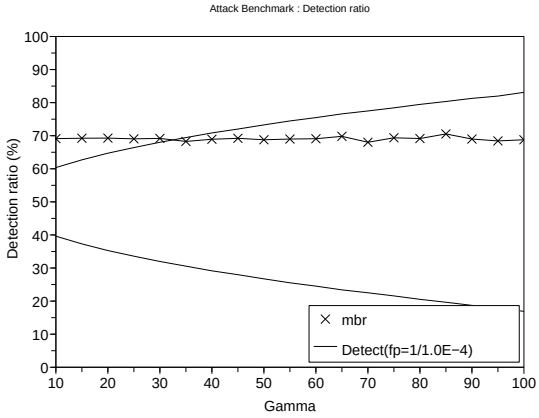


(f) Accuracy alteration

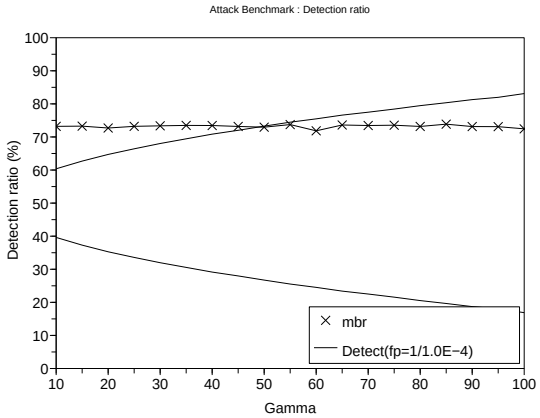
Figure 7.19: Change elongation filter



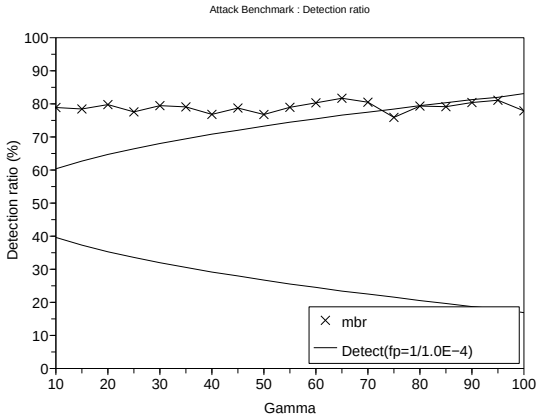
(a) WM



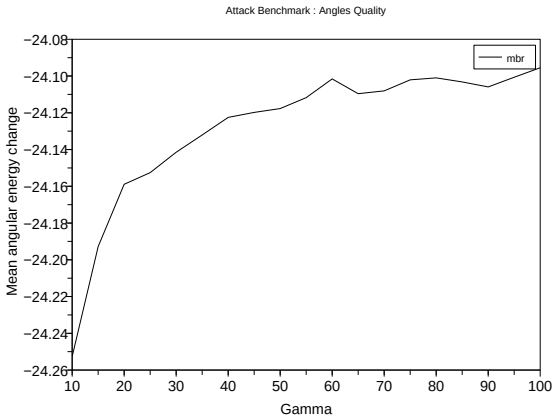
(b) RNW1



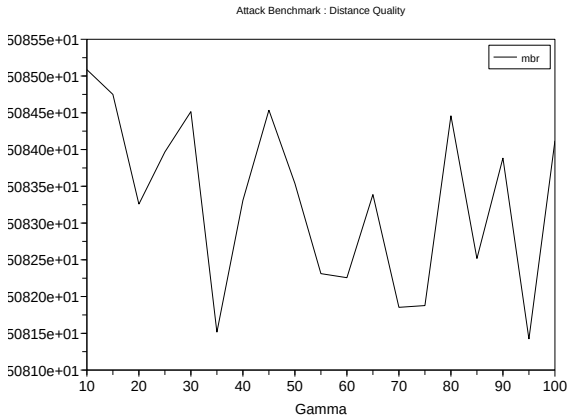
(c) RWN2



(d) Combined



(e) Angular quality



(f) Accuracy alteration

Figure 7.20: Minimum bounding rectangle filter

8

Conclusion & perspectives (English)

The work presented here is a crossover between two fields, watermarking and databases. In this conclusion I will first present some direct possible extensions: obtaining a richer constraint language and exchange model. Then I will discuss on the impact of databases methods, mainly their logical formalization, for classical watermarking theory. Finally I will conclude on the relationships between watermarking and databases in the Web era.

Natural extensions

Query-preservation: more expressive constraint languages

Up to now, the constraint-preservation methods we developed are mostly aggregate sum queries on a vocabulary that excludes numerical values. We consider here various extensions. First, the method of compensating alterations should extend to various aggregates and functions like statistical ones. Indeed, as long as the target function has a controllable variation when we alter its parameters, compensation can be applied. Some care is nevertheless needed when the precision of numbers comes into play.

Second, we would like to relax the constraint language so that altered values appear in the constraint itself (for example, watermarking a set of prices so that the number of equal prices remains the same). This extension is probably easy for constants in the language vocabulary. It may require supplementary hypothesis on data sets for general numerical values, as now the query to be preserved can assert properties on the numerical data.

It is also tempting to parameterize watermarking methods for geographical data with a spatial constraint language, like first-order logic on the real numbers with addition and order [55, 63]. For example, one may express that building watermarks should preserve their total surface. Again, for surfaces, compensation techniques apply. But it is well known that metrical aspects of spatial data are evading closed logical description in general [14].

Protocols: richer view model

In the presented model, the detector has access to the entire set of tuples that participate into the computation of the query to preserve. A richer model is introduced in [61], where the detector accesses only the result of aggregate queries. This extension for the present work is a natural direction.

In an even more general setting, the detector may only access to derived views of the suspect data set. In this case, the detector would have to first rebuilt a pertinent data set from these views. This goal is probably out of reach, since answering queries using views is a known difficult problem even if views are given [80], which is not the case in an adversarial setting. Results in this direction could nevertheless be obtained in a more restricted frame, namely views of streaming data. In the streaming context, views can be modeled by finite state transducers, and these machines possess interesting learning properties [68]. By monitoring a suspect stream based on the owner stream, and by performing continuous changes on original data, the detector may learn the view.

Relationship between the database and watermarking fields

From watermarking: revisiting classics

In the present work, the focus was voluntarily made of database-specific aspects. But there is a legitimate need to transpose and evaluate well-known watermarking classics in the context of databases. We mention a few directions here:

- Side-information attacks: while existing work considers uniformly distributed data, an attacker may use well-known data distributions to defeat the detector.

- Oracle attacks[1]: a particularly hard situation is when the attacker has access to a detector with a correct key as a black box. The attacker knows when the watermark is detected, and also knows the exact detection score. This confers a gradient descent strategy to erase the watermark: the attacker simply alters data and checks if the detection score decreases. Assessing the reality of this attack for databases and devising countermeasures is a must.

- More sophisticated embedding methods: most of the presented algorithms consider substitution watermarking, where mark bits replace existing data. These methods have known limitations, and other embeddings like quantization methods are to be preferred [22]. Nonetheless, this latter method is used in our geographical watermarking scheme.

- Full blindness vs. data blindness: the existing query preserving watermarking schemes still require side information at detection time, in addition to the secret key. This amount could be reduced.

- And more: public-key watermarking, zero-knowledge watermarking, etc.

To watermarking: an approach for security proofs

There may be a case where database methods, or rather its logical formalization, can contribute. A long-standing goal in the watermarking field is the obtainment of complete proofs of watermarking protocols, as they exist in the cryptographic setting. It is sometimes assessed that known proofs of watermarking protocols are limited to specific algorithms and classes of attacks, and lead to an "arm race" between markers and attackers. A better situation is a proof that any successful attacker would resolve an NP-complete problem efficiently or break some commonly accepted cryptographic assumption.

A small step in this direction was discussions with the SCALP project[2], that aims at proving cryptographic protocols using a proof assistant like Coq [15]. We obtained with Pierre Coutieu, Julien Lafaye, Philippe Audebaud and Xavier Urbain a rather limited proof of the Agrawal and Kiernan watermarking protocol, which lacked of generality.

Recent papers have proposed frameworks for *strong* watermarking proofs [51]. The method rely on the abstraction of watermarked data as a metric space with controlled properties. But such abstractions are often considered unrealistic [70]. The formalization obtained in Khanna and Zane's work [61] and the present one can give some insights in this direction. Indeed, modeling the distance between database instances under constraints has at least a clear and formal definition, which is not directly the case for multimedia documents. Obtaining a strong watermarking method for a simple case, for example graphs with a very specific similarity metric would be a first result. A glimpse in this direction was obtained by Julien Lafaye who considered the computational difficulty of computing key parameters of a watermarking method, when the similarity metric is given as a program. He has shown that this is respectively NP-hard for metrics defined by matrices, and EXP-hard on metrics defined by circuits.

[1] No, I do not mean Oracle's response to the IBM Almaden watermarking scheme.

[2] `http://scalp.gforge.inria.fr/`

(Watermarked) Data on the Web

As a conclusion to this conclusion, it is time to take a wider perspective, and – of course – to talk about the Web. On the one side, databases on the Web are facing semi-structured data formats, navigational query languages, and large-scale distributed computations to name a few examples. On the other side, the Web allows any user to become a content provider by using forums, blogs, twits, social networks, and so on. Sophisticated on-line contents can be elaborated by combining data from various sources and Web-service calls. In these scenarios, users may require (some) right management methods to assess intellectual rights on personnal productions. Hence, there is a natural need for scalable digital right management platforms that are able to broadcast on-demand content with personalized watermarks (see for example [58]).

Back to the database perspective, I would like to address the following questions:

- How to integrate intellectual property tools in flows of Web documents, that are dedicated to exchange, transformations and combinations with other documents during their life-time.
- How to specify intellectual rights policies for incoming and outgoing documents, and deploy them in a scalable and trustable way.

This year seconding at the WebDam project[3] is certainly a good starting point for this topic.

[3] `http://webdam.inria.fr`

9

Conclusion & perspectives (French)

Ce travail a présenté une hybridation entre deux domaines, le tatouage et les bases de données. Dans cette conclusion je présente quelques extensions possibles : enrichir le langage de contraintes ou le modèle d'échange des données. Ensuite je discuterai de l'impact des méthodes issues des bases de données, en particulier de leur formalisation logique, pour la théorie classique du tatouage. Enfin, je conclurai sur les relations entre tatouage et bases de données dans le contexte du Web.

Extensions naturelles

Préservation de requêtes : des langages de contraintes plus expressifs

Jusqu'à présent, les méthodes de préservation de contraintes que nous avons développées portent principalement sur les requêtes d'agrégat de somme sur un vocabulaire qui exclue les valeurs numériques. Nous considérons ici différentes extensions. Tout d'abord, la méthode des paires de compensation doit s'étendre à d'autre agrégats ou fonctions, comme celles issues de la statistique. En effet, du moment que la fonction cible possède une variation contrôlée lorsque l'on modifie ses paramètres d'entrée, la compensation peut être appliquée. Quelques précautions doivent cependant être prises si la précision des nombres vient à jouer un rôle.

Ensuite, nous voudrions permettre au langage de contraintes de manipuler les données numériques elles-mêmes (par exemple, tatouer un ensemble de prix de telle façon que le nombre de prix identiques reste le même). Cette extension est probablement facile pour l'ajout de constantes dans le langage. Des hypothèses supplémentaires sur les données seront probablement nécessaires dans un cas général, car alors, l'altération des valeurs numériques modifie les ensembles définis par les requêtes.

Il est également tentant de paramétrer notre algorithme de tatouage de données géographiques avec un langage de contraintes spatial (géométrique), comme la logique du premier ordre sur les nombres réels avec addition et ordre [55, 63]. Par exemple, on souhaiterait exprimer que le tatouage doit préserver la surface totale d'un bâtiment. Si l'on se restreint à de telles surfaces, la méthode des paires de compensation se généralise. Mais il est bien connu que les propriétés métriques des données géométriques échappent à une caractérisation logique en général [14].

Protocoles : enrichir le modèle de vues

Dans le modèle actuel, le détecteur a accès à l'ensemble des n-uplets qui participent au calcul de la requête à préserver. Un modèle plus sophistiqué et susceptible d'être rencontré en pratique a été proposé dans [61], où le détecteur n'accède qu'au résultat des requêtes. Réaliser cette extension aux résultats présentés ici est une direction naturelle.

Dans un cadre encore plus général, le détecteur n'a accès qu'à des vues dérivées des données suspectes. Le détecteur doit alors convertir les données issues de ces vues dans le format initialement souhaité. Cet objectif est probablement hors de portée, car répondre à une requête à partir de vues est un problème jugé difficile même si les vues sont explicitement décrites [80], ce qui n'est pas le cas dans un modèle avec

adversaire. Certains résultats pourraient cependant être obtenus dans un cadre plus restreint, par exemple les vues sur des données en flux. Pour celles-ci, les vues doivent être modélisées par des transducteurs finis, et ces machines possèdent d'intéressantes propriétés d'apprentissage [68]. En inspectant continuellement un flux suspect tout en faisant varier le flux tatoué, le détecteur pourrait alors inférer la vue utilisée.

Relations entre bases de données et tatouage

Apports du tatouage : les classiques

Dans ce travail, l'accent est mis sur les aspects du tatouage qui sont spécifiques aux bases de données. Mais il est également naturel de transposer et d'évaluer l'ensemble des résultats classiques du tatouage dans un contexte de bases de données. Nous en mentionnons quelque-uns :

- Attaquant informé : alors que les travaux existants supposent une distribution uniforme des données, un attaquant peut tirer partie d'une distribution connue a priori pour vaincre le détecteur.
- Attaque par Oracle[1] : une situation particulièrement délicate est quand l'attaquant a accès à un détecteur avec sa clé privée sous forme d'une boite noire. L'attaquant sait alors quand le tatouage est détecté, et connait également le score de détection. Cette configuration permet à l'attaquant de déployer une attaque par descente de gradient pour effacer le tatouage : l'attaquant altère une partie des données et vérifie si le score de détection diminue. Etablir des contre-mesures pour ce type d'attaque pour les bases de données est un impératif.
- Encodage du tatouage plus sophistiqué : la plupart des algorithmes présentés réalise un tatouage par substitution, où les bits de marque remplacent les données existantes. Ces méthodes ont des limitations connues, et ont leur préfère en général d'autres approches comme la quantification [22]. Nous utilisons cependant cette dernière méthode dans notre algorithme de tatouage de données géographiques.
- Protocole purement aveugle ou aveugle aux données : les méthodes connues de tatouage avec préservation de requêtes nécessitent de l'information supplémentaire lors de la détection, en supplément de la clé secrète. Cette quantité d'information devrait être réduite.
- Et plus : tatouage à clé publique, à divulgation nulle, etc.

Apports au tatouage : techniques de preuve de sécurité

Dans certains cas, les méthodes issues de bases de données, où plutôt de leur formalisation logique, peuvent contribuer aux connaissances sur le tatouage en général. Un des objectifs à long terme du tatouage est l'obtention de preuves complètes de sécurité, dans un esprit similaires aux preuves des protocoles cryptographiques. Il est parfois affirmé que les preuves existantes sont limitées à des algorithmes et des classes d'attaques spécifiques, et conduisent inévitablement à une « escalade » entre tatoueurs et attaquants. Une meilleure situation serait d'obtenir une preuve de la forme suivante : tout attaquant victorieux doit avoir résolu un problème NP-complet efficacement, ou avoir transgressé une hypothèse cryptographique communément admise.

Un essai dans cette direction ont été des discussions avec les membres du projet ANR SCALP[2], dont le but est de certifier les preuves de protocoles cryptographiques en utilisant un assistant de preuve comme Coq [15]. Nous avons obtenu avec Pierre Coutieu, Julien Lafaye, Philippe Audebaud et Xavier Urbain une preuve (plutôt restreinte) du protocole d'Agrawal et Kiernan.

Des travaux récents proposent de nouveaux cadres pour des preuves *fortes* de protocoles de tatouage [51]. La méthode repose sur une abstraction des données à tatouer en un espace métrique avec de bonnes propriétés. Mais ces abstractions sont par certains considérées comme non-réalistes [70]. Les formalisations

[1] Non, il ne s'agit pas de la réponse d'Oracle au schéma de tatouage proposé par IBM Almaden.

[2] `http://scalp.gforge.inria.fr/`

proposées par Khanna et Zane [61] ainsi que celles de ce document peuvent fournir quelques éclairages dans cette direction. En effet, modéliser la distance entre instances de bases de données sous contraintes a au moins un définition précise et formelle, ce qui n'est pas le cas directement pour les documents multimédia. Obtenir une preuve forte d'un protocole de tatouage pour une métrique de similarité ad hoc serait un premier résultat. Un pas dans cette voie a été obtenu par Julien Lafaye qui a étudié la difficulté de calculer des paramètres intéressants d'une problème de tatouage, lorsque la métrique de similarité est donnée comme un programme. Il a montré que ces calculs sont NP-difficiles pour les métriques définies par matrices, et EXP-difficiles pour celles définies par circuits.

Données (tatouées) sur le Web

En conclusion de cette conclusion, il est temps – évidemment – de parler un peu du Web. D'une part, les bases de données, confrontées aux Web, ont intégré les données semi-structurées, les langages d'interrogations navigationnels, les calculs de requêtes massivement distribués, pour citer quelques aspects. D'autre part, le Web permet à n'importe quel utilisateur de devenir un fournisseur de contenu, par l'utilisation de forums, blogs, twits, réseaux sociaux, etc. Des contenus en ligne sophistiqués peuvent ainsi être déployés par combinaison de données de sources diverses et d'appels de services. Dans ces scénarios, les utilisateurs peuvent avoir besoin de (quelques) méthodes de protection de la propriété intellectuelle pour leurs productions personnelles. Ceci motive la création de plateforme de tatouage passant à l'échelle capable de diffuser du contenu à la demande avec des tatouages individualisés (voir par exemple [58]). En reprenant le point de vue des bases de données, j'aimerais aborder les questions suivantes :

- Comment intégrer les outils de protection de la propriété intellectuelle dans un flot de documents Web, qui sont par nature dédiés à l'échange, la transformation et la combinaison avec d'autres documents durant leur cycle de vie.
- Comment spécifier des politiques de gestion des droits intellectuels pour les documents entrant et sortant d'un système, et comment les appliquer à grande échelle et en toute confiance.

Cette année de délégation INRIA dans le projet WebDam[3], dédié à la gestion de données sur le Web, est certainement un bon point de départ pour ces questions.

[3] `http://webdam.inria.fr`

A
Other studies

A.1 Work presented in this report

Journals

- **Julien Lafaye, David Gross-Amblard, Camélia Constantin and Meryem Guerrouani.** WATERMILL**: an optimized fingerprinting system for highly constrained data.** ***IEEE Transactions on Knowledge and Data Engineering (TKDE)*****, 20(4): 532-546, April 2008.**

- **David Gross-Amblard. Query-Preserving Watermarking of Relational Databases and XML Documents. To appear in** ***ACM Transactions on Database Systems (ACM TODS)*****, 36(1), 2010 (tentatively scheduled).**

Conferences

- **David Gross-Amblard, Philippe Rigaux, Lylia Abrouk and Nadine Cullot. Fingering watermarking in symbolic digital scores. In** ***International Conference on Music Information Retrieval (ISMIR)*****, 2009, Kobe, Japan.**

- Julien Lafaye, Jean Béguec, David Gross-Amblard and Anne Ruas. Invisible Graffiti on your Buildings: Blind & Squaring-proof Watermarking of Geographical Databases. In *10th International Symposium on Spatial and Temporal Databases (SSTD)*, July 16-18, 2007, Boston. LNCS 4605, pages 312-329.

- Julien Lafaye and David Gross-Amblard. XML Streams Watermarking. In *20th Annual IFIP WG 11.3 Working Conference on Data and Applications Security (DBSec2006)*, Sophia Antipolis, France, 7/31 - 8/02 2006, pages 74–88.

- **David Gross-Amblard. Query-Preserving Watermarking of Relational Databases and XML Documents. In** ***ACM Principles of Database Systems (PODS)*****, 2003, pages 191–201.**

Workshops

- Camélia Constantin, David Gross-Amblard and Meryem Guerrouani. Watermill: an Optimized Fingerprinting Tool for Highly Constrained Data. In *ACM Workshop on Multimedia and Security (MMSec)*, New-York, USA, August 1-2 2005, pp. 143-155, 2005.

Softs

- (soft) Camélia Constantin, David Gross-Amblard, Meryem Guerrouani et Julien Lafaye. WATERMILL: an optimized watermarking/fingerprinting tool for databases.
  `http://watermill.sf.net`

A.2 Miscellaneous

A.2.1 Multimedia watermarking

In this work, a general framework for watermarking multimedia documents is proposed. A multimedia document is seen as a relational structure between components (for example images and polygonal annotations on them), along with functional dependencies. The framework guarantees that the watermarking of one component is consistent with its dependencies.

Conferences

- Richard Chbeir and David Gross-Amblard. Multimedia and Metadata Watermarking Driven by Application Constraints. In *IEEE Multi Media Modelling conference (MMM)*, 8 pp., 2006.

A.2.2 Other supervised work

In this section I also mention papers of my former PhD student, Julien Lafaye, whose thesis is entitled "Database watermarking under constraints" (defended 2007). Although I supervised his thesis, these works are Julien's own ideas.

Workshops

- Julien Lafaye. An analysis of database watermarking security. In *International Workshop on Data Hiding for Information and Multimedia Security (DHIMS)*, pages 462-467, Manchester, UK, 08/29 - 08/31 2007.

  This paper studies the security of Agrawal and Kiernan's scheme for database watermarking, elaborating on new tools from Cayre, Furon and Fontaine [20]. It shows that using high significant bits of numerical data sets as part of the secret key may lead to information leakage.

- Julien Lafaye. On the Complexity of Obtaining Optimal Watermarking Schemes. In *International Workshop on Digital Watermarking (IWDW)*, Guangzhou, China, 12/03-12/05, 2007.

  In this paper, Julien Lafaye studies the computational problem of assessing the robustness and false-positive rate of a watermarking method, described as a matrix or a circuit. It shows that this problem is NP-hard on matrices and EXP-hard on circuits.

Softs

- Julien Lafaye and Jean Béguec. The geographical database watermarking library WATERGOAT (OpenJump module).
  `http://cedric.cnam.fr/~lafaye_j/index.php?n=Main.WaterGoatOpenJumpPlugin`

A.3 Work non-related to watermarking

A.3.1 Web services ranking

Web services are a normalized W3C technology allowing Web servers to expose portions of their code to users, instead of basic (static) Web pages. A wide amount of such services appeared recently, and there is a crucial need for their indexing and ranking. Traditional solutions are based on open registries (UDDI), where services types and descriptions are provided. If such descriptions are not available, no longer up-to-date or simply not sufficient for ranking, we have proposed a Web service ranking method derived from Google's PageRank method. A Web service is seen as a Web page, while a service call from one service to another is considered as a link between two pages. This approach, including time-dependencies, was part of Camélia Constantin's thesis.

Conferences

- **Camélia Constantin, Bernd Amann, David Gross-Amblard. A Link-Based Ranking Model for Services. In *Cooperative Information Systems (CoopIS) International Conference, 2006*, pages 327-344.**

National journals

- Camelia Constantin, Bernd Amann and David Gross-Amblard. Un modèle de classement de services par contribution et utilité. In *Revue des sciences et technologies de l'information* (numéro special "Recherche d'information dans les systemes d'information avances") (1633-1311) - 12(1) : 33-60, 2007.

A.3.2 Web publishing-by-example

Publishing data on the Web by dynamically extracting content from a database is nowadays a common practice (with tools like Apache, Mysql, PHP, Python, etc.). However these techniques are limited to users with reasonable programming skills. To allow natural users of blogs or wikis to access such data sets, or to increase programmers productivity, we have proposed a publish-by-example model. In this setting, the system extracts from the database or its schema a canonical database of examples. By building template Web pages with such examples, users can obtain automatically a full publishing program that generalized for a whole Website. The main point is to guarantee that the example data set is rich enough to express all interesting queries (Sonia Guéhis'thesis and following publications).

Conferences

- Sonia Guéhis, David Gross-Amblard and Philippe Rigaux. Publish By Example. In *IEEE International Conference on Web Engineering (ICWE)*, July 14-18, 2008, Yorktown Heights, New York.

National journals

- Sonia Guéhis, David Gross-Amblard, Philippe Rigaux. Un modèle de production interactive de programmes de publication. *Ingénierie des Systèmes d'Information (Networking and Information Systems), revue des sciences et technologies de l'information (RTSI) série ISI*, 13 (5) : 107-130, octobre 2008.

Softs

- Sonia Guéhis. The DocQL publication suite.
`http://www.lamsade.dauphine.fr/~guehis/docql/`

A.3.3 Time-series management

Time-series is a key concept to handle useful information flows: environmental monitoring by sensors, stock exchange, news articles, and so forth. Along with Zoé Faget, Virginie Goasdoué-Thion and Philippe Rigaux, we proposed a query language and an algebra that manipulates time-series. Our main application is the management of musical events into the NEUMA project.

Conferences

- Zoé Faget, David Gross-Amblard, Philippe Rigaux, Virginie Thion-Goasoué. Modeling Synchronized Time Series. In *International Database Engineering & Applications Symposium (IDEAS)*, Montreal, QC, Canada , August 2010.

Workshops

- Lylia Abrouk, Hervé Audéon, Nadine Cullot, Cécile Davy-Rigaux, Zoé Faget, David Gross-Amblard, Hyunja Lee, Philippe Rigaux, Alice Tacaille, Elisabeth Gavignet, Virginie Thion-Goasdoué. The NEUMA Project: Towards Cooperative On-line Music Score Libraries. In *Workshop on Exploring Musical Information Spaces (WEMIS)*, Corfu, Greece, 2009.

B
Résumé d'activité

B.1 Encadrement (thèses, postdocs, masters, ingénieurs)

Thèses

1 (en cours) Encadrement (33%) avec Lylia Abrouk (33%) et Christophe Nicolle (33%) de la thèse de Damien Leprovost (boursier Jeune chercheur entrepreneur–JCE Conseil régional de Bourgogne), intitulée "Découverte de communautés par analyse sémantique des usages", débutée en septembre 2009.

2 Encadrement (95%) avec Michel Scholl (5%) de la thèse de Julien Lafaye (boursier AMX), intitulée "Tatouage des bases de données avec préservation de contraintes", débutée en septembre 2004, soutenue le 7 novembre 2007 (durée 3 ans, 2 mois). Julien est qualifié en 27e section et actuellement salarié de la SSII Scimetis.

3 Encadrement (30%) avec Bernd Amann (70%) de la thèse de Camélia Constantin (boursière MENRT), intitulée "Classement de services par utilité", débutée en septembre 2004, soutenue le 27 novembre 2007 (durée 3 ans, 2 mois). Camélia est actuellement maître de conférences à l'Université Paris VI et au LIP6.

Postdocs

1 (en cours) Encadrement (50%) avec Lylia Abrouk (50%) du stage post-doctoral de Chi Dung Tran, dans le cadre de l'ANR NEUMA, portant sur les aspects collaboratifs des bases de données musicales, de septembre 2010 à septembre 2011.

2 Encadrement avec Lylia Abrouk, Nadine Cullot et Elisabeth Gavignet du stage post-doctoral de Hyunja Lee, dans le cadre de l'ANR NEUMA, portant sur la construction d'une ontologie musicale, de mai 2009 à mai 2010.

Masters avec résultats en recherche

1 Encadrement du master 3i / recherche (Université de Bourgogne) d'Émilien Antoine, intitulé "Tatouage de partitions musicales symboliques par altération sémantique d'accords", de mars 2010 à juin 2010 (noté 16/20, actuellement en thèse à l'INRIA sous la direction de Serge Abiteboul).

2 Encadrement du master VIIAM multimédia / professionnel (Université de Bourgogne) de Damien Leprovost, intitulé "Liaison OpenOffice / Beamer", de mars 2009 à juin 2009 (noté 17/20, actuellement en thèse sous ma co-direction).

3 Co-encadrant (25%), avec Anne Ruas (50%) et Julien Lafaye (25%) du master Science de l'information géographique / professionnel (Université de Marne-la-Vallée/Ecole nationale des sciences géographiques) de Jean Béguec, intitulé "Diffusion et Tatouage des Données Géographiques", de avril 2006 à septembre

2006 (noté 17/20, actuellement employé par la société Capgemini). Résultats du stage publié dans la conférence SSTD'2007 [65].

4 Encadrant (50%) avec Anne Ruas (50%) du master SAR/recherche (Université Pierre et Marie Curie - Paris 6) de Ammar Mechouche, intitulé "Tatouage de données géographiques", de avril à septembre 2005 (noté 18/20, meilleure note de stage du master, thèse effectuée à l'IRISA, actuellement en postdoc à l'IGN).

5 Encadrant (100%) du master I3/recherche (Paris XI-Orsay) de Julien Lafaye, intitulé "Enhancing security of Web Services Workflows using Watermarking", de mars à juin 2004 (noté 16/20, a soutenu une thèse sous ma direction).

6 Encadrant (50%) avec Bernd Amann (50%) du master SAR/recherche (UPMC) de Camelia Constantin, intitulé "Calcul d'importance de services Web", de mars à juin 2004 (notée 18/20). Stage ayant donné lieu à publication en conférence [24] et à la revue des sciences et technologies de l'information [25]. A soutenu une thèse sous ma co-direction.

Mémoire d'ingénieur et fin d'étude

1 Mémoire d'ingénieur de Camélia Constantin, École polytechnique de Bucarest, intitulé "Tatouage sous contraintes", de avril à juin 2003. Stage ayant donné lieu à publication [26]. A soutenu un thèse sous ma direction (encadrant principal, Bernd Amann), avec financement MENRT.

2 Mémoire d'ingénieur Cnam de Meryem Guerrouani, intitulé "Tatouage de documents XML contraints", de septembre 2003 à juin 2005. Notée 18/20, stage ayant donné lieu à publication en conférence [26] et revue [67], actuellement employée dans une SSII.

3 Différents autres mémoires d'ingénieur Cnam : Guillaume Chalade (2006), Karine Volpi (2006), Robert Abo (2006), Mai Hoa Guennou (2007).

B.2 Animation scientifique

Projets et financements

PEPS CNRS STRATES Ce PEPS, obtenu en avril 2010, pour un montant de 10 kE, propose une approche inter-disciplinaire pour étudier les modèles économiques des données. Il est mené un collaboration avec un mathématicien-économiste du CEREMADE (Université de Dauphine), et deux économistes de l'École d'économie de Paris. J'ai mené le montage de ce projet et en suis le coordinateur (taux d'acceptation inférieur à 1/10).

ANR Contenu et interaction NEUMA Cette ANR, obtenue fin 2008 pour une période de 3 ans et pour un montant de 620 kE, s'intéresse au partage, au stockage et à l'interrogation de grands corpus de données musicales sous forme de partitions. Ce projet, mené en collaboration avec des musicologues du CNRS (Institut de recherche sur le patrimoine de musical en France - IRPFM), croise des techniques de bases de données pures (stockage, interrogation) avec une approche contemporaine du Web (Web collaboratif pour le partage) et de la sécurité (sécurisation des partitions par tatouage). Ce projet est coordonnée par le LAMSADE (Université de Paris-Dauphine), en partenariat avec le Le2i (Université de Bourgogne), l'IRPMF (CNRS), et une entreprise de gestion de contenu multimédia, ARMADILLO.

ACI Sécurité & Informatique Tadorne J'ai mené le montage de ce projet (rencontre avec les différents intervenants, centralisation et finalisation de la proposition de projet). J'ai été coordinateur de cette ACI obtenue pour la période 2005-2007, pour un montant total de 62 kE. Le projet Tadorne (tatouage de données contraintes [4]) s'est attaqué au problème du tatouage de données et services sous contraintes, en prenant pour application cible les données et services géographiques de l'institut géographique

national (IGN). Cette ACI a rassemblé huit chercheurs de différents laboratoires (CEDRIC/CNAM, GREYC/université de Caen, LAMSADE/université Paris-Dauphine, COGIT/IGN).

BQR Codes d'estampillages incrémentaux Ce BQR a été obtenu en 2007 en collaboration avec l'équipe Algorithmique combinatoire du laboratoire Le2i, pour un montant de 6 kE.

BQR Certification d'algorithme de tatouage Ce BQR a été obtenu en 2006 en collaboration avec l'équipe Conception & programmation raisonnée du laboratoire Cédric, pour un montant de 7,7 kE.

Collaboration nationales

- **Délégation INRIA à partir de septembre 2010, dans le projet WebDam ERC Grant de Serge Abiteboul.**
- Membre extérieur du PPF Wisdom depuis janvier 2007 (Cnam-Paris, université de Dauphine, université Paris VI), `http://wisdom.lip6.fr/`, et collaboration particulières avec Bernd Amann (Professeur, LIP6) et Philippe Rigaux (Professeur, LAMSADE).
- Participant extérieur aux ACI SemWeb et SCALP.
- Collaboration avec Cristina Bazgan (professeur à l'université de Dauphine, dans le cadre de l'ACI Tadorne).
- Collaboration avec Yannick Viossat (CEREMADE, maître de conférences, Université Paris-Dauphine) et Gabrielle Demange (directrice d'étude, École d'économie de Paris).
- Collaboration avec l'Institut géographique national (Anne Ruas, directrice du laboratoire Cogit).

Interventions en séminaires de recherche de haut niveau

- Tatouage de données contraintes, journées Codage et cryptographie (C2), Aussois, 30 janvier au 4 février 2005.
- Présentation du projet Tadorne, journées des ACI Sécurité & Informatique 2004, LAAS, Toulouse, du 15 au 17 novembre 2004.
- Tatouage de bases de données, séminaire Cryptographie, codage et algorithmique (CCA), ENSTA, 16 janvier 2004.

B.3 Relations avec le monde industriel ou socio-économique

J'ai participé à l'expertise de deux dossiers du réseau national des technologies logicielles (RNTL) en 2003, et un dossier ANR avec partenariat industriel en 2006, portant sur la sécurité. Je participe à l'ANR 2008 Contenu et Interaction NEUMA, classée Recherche Industrielle, en partenariat avec l'entreprise de gestion de contenu multimédia ARMADILLO. J'ai également réalisé une prestation de conseil technologique (PCT) auprès d'une entreprise de nouvelles technologies en Bourgogne, en partenariat avec UB-Filliale Wellience.

B.4 Visibilité

Participation en jury de thèses

- (Comme examinateur) Sonia Guéhis, le 2 décembre 2009.

- (Comme co-directeur) Julien Lafaye, soutenance le 7 novembre 2007, jury composé de Marie-Christine Costa (présidente), Alban Gabillon, Jerry Kiernan (rapporteurs), Serge Abiteboul, , Cristina Bazgan (examinateur), Michel Scholl (co-directeur).
- (Comme co-directeur) Camélia Constantin, soutenance le 27 novembre 2007, jury composé de Christine Collet (présidente et examinatrice), Michalis Vazirgiannis (rapporteur), Serge Abiteboul, Michel Scholl (examinateurs), Bernd Amann (co-directeur).

Invitation

- Tatouage de bases de données. Cours invité. École thématique BDA Masses de données distribuées, École de physique des Houches, 16-21 mai 2010.
- Présentation invitée au Workshop intenational PresDB 2007 (International Workshop on Databases Preservation, Edinbourgh, March 23, 2007), intitulée "Database watermarking: protection by alteration".

Comité de programme

- Comité scientifique de l'école thématique BDA - Masses de données distribuées, Les Houches, 16-21 mai 2010, organisé par Nicole Bidoit (LRI/Université Paris Sud 11) et Philippe Pucheral (PRISM/INRIA Rocquencourt/Université Versailles Saint-Quentin).
- Membre du comité de programme des conférences internationales CSTST 2008, ICDIM 2008 et du workshop étudiant MEDES-SW 2009.
- Président et membre du comité de sélection des démonstrations pour la conférence nationale Bases de données avancées (BDA) 2008.
- Président et membre du comité de programme de l'atelier international SWAN (1st Workshop on Security and Trust of Web-oriented Application Networks), Bangalore, 6-8 décembre 2006.
- Membre du comité de programme des conférences nationales Bases de données avancées (BDA) 2005, 2008, 2009 et 2010 (dont session de démonstration).
- Relecteur pour les revues JCSS (2005), TKDE (2005, 2006), Information systems (2007), TDSC (2005), TISSEC (2005), WWWJournal (2005), Acta Informatica (2005), Infosec (2004) et TODS (2003).
- Relecteur secondaire pour les conférences ACM SIGSPATIAL GIS 2009, ACNS 2007, ASIACCS 2007, ICDE 2007, ICDIM 2006 et 2007, ASIAN 2005, PODS 2005, SOFSEM 2005, VLDB 2005, EDBT 2004, VLDB 2003.

Commissions

En 2010 Membre extérieur des CdS de Polytech/LIRMM Montpellier et de l'ESIAL/LORIA Nancy.

Depuis septembre 2008 Membre suppléant du bureau de la commission de proposition section 27 du laboratoire LE2I UMR CNRS 5158 (université de Bourgogne).

De septembre 2006 à septembre 2008 Membre suppléant de la commission de spécialiste section 27 du laboratoire LE2I UMR CNRS 5158 (université de Bourgogne).

De janvier 2005 à septembre 2006 Membre du bureau de direction du laboratoire CEDRIC (4 membres), responsable des relations avec la communauté (CNU, Spécif, etc.) ainsi que de la commission Web.

De janvier 2004 à septembre 2006 Membre extérieur titulaire de la commission de spécialiste section 27 du laboratoire LE2I UMR CNRS 5158 (université de Bourgogne).

De janvier 2001 à septembre 2006 Membre titulaire du conseil du laboratoire CEDRIC, suppléant depuis 2005.

Primes

- Titulaire de la PEDR depuis septembre 2007.

Formations en relation avec le domaine de recherche

- (depuis juin 2009) Formation "Tatouage des bases de données", École nationale de la statistique et de l'analyse de l'information (ENSAI), Campus de Ker Lann, Rennes.
- Voir également la section "Invitation".

B.5 Activité d'enseignement

Activités statutaires J'ai enseigné l'informatique à tous les niveaux du L1 au M2 professionnel ou recherche, selon les quantités suivantes :

- 2001-2002 : 214 ETD
- 2002-2003 : 218 ETD
- 2003-2004 : 192 ETD
- 2004-2005 : 234 ETD
- 2005-2006 : 208 ETD
- 2006-2007 : 254 ETD
- 2007-2008 : 201 ETD
- 2008-2009 : 210 ETD
- 2009-2010 : 220 ETD

Responsabilités pédagogiques

- (L1-depuis 2008) Responsable du module Programmation objet en Java au niveau L1
- (M2-depuis 2006) Co-responsable du module de Systèmes d'information multimedia du master professionnel VIIAM, avec Lylia Abrouk
- (M1-2004-2005) Co-responsable du module Bases de données et Web en valeur C (niveau Master 1) du Cnam-Paris, avec Dan Vodislav
- (L3-2004-2005) Co-responsable du cours de bases de données en valeur B (niveau L3) en 2004-2005 du Cnam-Paris, avec Dan Vodislav

Thématiques abordées

- Théoriques : théorie des langages, compilation, programmation objet et fonctionnelle, théorie des bases de données, optimisation combinatoire, algorithmique.
- Appliqué : optimisation des bases de données, langage SQL, langages du Web (XML, DTD, XML Schema, XPath, XSLT, Services Web), langages de programmation (Java, Ocaml, Pascal, PHP), Système et réseaux, architecture des machines, systèmes distribués et parallèles (MPI).

Bibliography

[1] CNN RSS `http://www.cnn.com/services/rss/`.

[2] GéoPortail (visited 20/10/2006). `http://www.geoportail.fr`.

[3] OpenJUMP. `http://openjump.org`.

[4] Projet Tadorne (tatouage de données contraintes). `http://cedric.cnam.fr/vertigo/tadorne`.

[5] `http://kdd.ics.uci.edu/`.

[6] The NEUMA Project. `http://neuma.irpmf-cnrs.fr`.

[7] Rakesh Agrawal, Peter J. Haas, and Jerry Kiernan. Watermarking Relational Data: Framework, Algorithms and Analysis. *VLDB J.*, 12(2):157–169, 2003.

[8] Rakesh Agrawal and Jerry Kiernan. Watermarking Relational Databases. In *International Conference on Very Large Databases (VLDB)*, 2002.

[9] Rakesh Agrawal, Jerry Kiernan, Ramakrishnan Srikant, and Yirong Xu. Hippocratic databases. In *VLDB*, pages 143–154. Morgan Kaufmann, 2002.

[10] Sylvain Airault. De la base de données à la carte : une approche globale pour l'équarrissage de bâtiments (in french). *Revue Internationale de Géomatique*, 6(2-3):203–217, 1996.

[11] Alia Al Kasimi, Eric Nichols, and Christopher Raphael. A Simple Algorithm for Automatic Generation of Polyphonic Piano Fingerings. In *ISMIR*, pages 355–356, 2007.

[12] Arvind Arasu, Brian Babcock, Shivnath Babu, Mayur Datar, Keith Ito, Rajeev Motwani, Itaru Nishizawa, Utkarsh Srivastava, Dilys Thomas, Rohit Varma, and Jennifer Widom. STREAM: The Stanford Stream Data Manager. *IEEE Data Engineering Bulletin*, 26(1):19–26, 2003.

[13] Cyril Bazin, Jean-Marie Le Bars, and Jacques Madelaine. A blind, fast and robust method for geographical data watermarking. In *Proceedings of the 2nd ACM symposium on Information, Computer and Communications Security, ASIASCCS'07*, pages 265–272, March 2007.

[14] Michael Benedikt and Leonid Libkin. Exact and approximate aggregation in constraint query languages. In *Symposium on Principles of Databases Systems (PODS)*, pages 102–113, 1999.

[15] Yves Bertot and Pierre Castéran. *Interactive Theorem Proving and Program Development. Coq'Art: The Calculus of Inductive Constructions*. Texts in Theoretical Computer Science. Springer Verlag, 2004.

[16] Anselm Blumer, Andrezj Ehrenfeucht, David Haussler, and Manfred K. Warmuth. Learnability and the Vapnik-Chervonenkis dimension. *Journal of the Association of Computing Machinery (JACM)*, 36(4):929–965, October 1989.

[17] D. Boneh and J. Shaw. Collusion-secure fingerprinting for digital data. *IEEE Transactions of Information Theory*, 44, 1998.

[18] Peter Buneman, Sanjeev Khanna, and Wang Chiew Tan. Why and where: A characterization of data provenance. In Jan Van den Bussche and Victor Vianu, editors, *Database Theory - ICDT 2001, 8th International Conference, London, UK, January 4-6, 2001, Proceedings*, volume 1973 of *Lecture Notes in Computer Science*, pages 316–330. Springer, 2001.

[19] V. Cappelini, F. Bartolini, and M. Barni. Information theoretic aspects in digital watermarking. *Signal Process.*, 81(6):1117–1119, 2001.

[20] F. Cayre, C. Fontaine, and T. Furon. Watermarking security: Theory and practice. *IEEE Transactions on Signal Processing*, 53(10):3976–3987, 2005. special issue "Supplement on Secure Media III".

[21] Richard Chbeir and David Gross-Amblard. Multimedia and Metadata Watermarking Driven by Application Constraints. In *IEEE Multi Media Modelling conference (MMM)*, 2006.

[22] Brian Chen and Gregory W. Wornell. Quantization index modulation: A class of provably good methods for digital watermarking and information embedding. *IEEE Transactions on Information Theory*, 47(4):1423–1443, 2001.

[23] COGIT laboratory. Geoxygene: an open framework for the development and deployment of gis applications, 2007.

[24] Camélia Constantin, Bernd Amann, and David Gross-Amblard. A link-based ranking model for services. In Robert Meersman and Zahir Tari, editors, *OTM Conferences (1)*, volume 4275 of *Lecture Notes in Computer Science*, pages 327–344. Springer, 2006.

[25] Camélia Constantin, Bernd Amann, and David Gross-Amblard. Un modèle de classement de services par contribution et utilité. *Ingénierie des Systèmes d'Information*, 12(1):33–60, 2007.

[26] Camelia Constantin, David Gross-Amblard, and Meryem Guerrouani. Watermill: an Optimized Fingerprinting System for Highly Constrained Data. In *ACM MultiMedia and Security Workshop*, New York City, New York, USA, January 1–2 2005.

[27] Camelia Constantin, David Gross-Amblard, Meryem Guerrouani, and Julien Lafaye. Logiciel Watermill. `http://watermill.sourceforge.net`.

[28] Ingemar J. Cox, Matthew L. Miller, and Jeffrey A. Bloom. *Digital Watermarking*. Morgan Kaufmann Publishers, Inc., San Francisco, 2001.

[29] D. Douglas and T. Peucker. Algorithms for the reduction of the number of points required for represent a digitized line or its caricature. *Canadian Cartographer*, 10(2):112–122, 1973.

[30] Cécile Duchêne, Sylvain Bard, Xavier Barillot, Anne Ruas, Jenny Trevisan, and Florence Holzapfel. Quantitative and qualitative description of building orientation. In *Fifth workshop on progress in automated map generalisation, ICA, commission on map generalisation*, April 2003.

[31] Jörg Flum, Markus Frick, and Martin Grohe. Query evaluation via tree-decompositions. *Journal of the Association of Computing Machinery (JACM)*, 49:716–752, 2002.

[32] Wolfgang Funk and Martin Schmucker. High Capacity Information Hiding in Music Scores. In Paolo Nesi, editor, *First International Conference on WEB Delivering of Music, proceedings of Wedelmusic 2001*, number 5020 in IEEE Computer Society Technical Committee on Computer Generated Music, pages 12–19, Los Alamitos, California, USA, 2001. IEEE Computer Society.

[33] H. Gaifman. On local and non-local properties. In *Proceedings of the Herbrand Symposium, Logic Colloquium'1981*, North Holland, 1982.

[34] Georg Gottlob, Christoph Koch, and Reinhard Pichler. Efficient algorithms for processing xpath queries. *ACM Trans. Database Syst.*, 30(2):444–491, 2005.

[35] Erich Grädel and Yuri Gurevich. Metafinite model theory. *Inf. Comput.*, 140(1):26–81, 1998.

[36] Martin Grohe and Thomas Schwentick. Locality of order-invariant first-order formulas. *ACM Transactions on Computational Logic (TOCL)*, 1(1):112–130, 2000.

[37] Martin Grohe and György Turán. Learnability and definability in trees and similar structures. *Theory of Computing Systems (TOCS)*, 37:193–220, 2004.

[38] David Gross and Michel de Rougemont. Uniform Generation in Spatial Constraint Databases and Applications. In *Symposium on Principles of Databases Systems (PODS)*, pages 254–259, 2000.

[39] David Gross-Amblard. Query-Preserving Watermarking of Relational Databases and XML Documents. In *Symposium on Principles of Databases Systems (PODS)*, pages 191–201, 2003.

[40] David Gross-Amblard. Query-Preserving Watermarking of Relational Databases and XML Documents. *ACM Transactions on Database Systems (TODS)*, 36(1), 2010. To appear, tentatively scheduled.

[41] David Gross-Amblard and Michel de Rougemont. Uniform Generation in Spatial Constraint Databases and Applications. *Journal of Computer and System Sciences (JCSS)*, 72(4):576–591, June 2006. accepted November 2004, available on-line on ScienceDirect, `doi:doi:10.1016/j.jcss.2005.09.008`.

[42] David Gross-Amblard and Julien Lafaye. XML streams watermarking. Technical report, CEDRIC, 2005.

[43] David Gross-Amblard, Philippe Rigaux, Lylia Abrouk, and Nadine Cullot. Fingering watermarking in symbolic digital scores. In *International Conference on Music Information Retrieval (ISMIR)*, 2009.

[44] Stephane Grumbach, Leonid Libkin, Tova Milo, and Limsoon Wong. Query language for bags: expressive power and complexity. *SIGACT News*, 27:30–37, 1996.

[45] Sonia Guehis, David Gross-Amblard, and Philippe Rigaux. Publish by example. In Daniel Schwabe, Francisco Curbera, and Paul Dantzig, editors, *ICWE*, pages 45–51. IEEE, 2008.

[46] Sonia Guehis, David Gross-Amblard, and Philippe Rigaux. Un modèle de production interactive de programmes de publication. *Ingénierie des Systèmes d'Information*, 13(5):107–130, 2008.

[47] Hans-Jörgen Guth and Birgit Pfitzmann. Error- and Collusion-Secure Fingerprinting for Digital Data. In Andreas Pfitzmann, editor, *Information Hiding, Third International Workshop, IH'99*, number 1768 in Lecture Notes in Computer Science, pages 134–145. Springer, 2000.

[48] Melanie Hart, Robert Bosch, and Elbert Tsai. Finding optimal piano fingerings. *The UMAP Journal*, 2(21):167–177, 2000.

[49] Lauri Hella, Leonid Libkin, Juha Nurmonen, and Limsoon Wong. Logics with aggregate operators. *Journal of the Association of Computing Machinery (JACM)*, 48(4):880–907, 2001.

[50] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, March 1963.

[51] Nicholas Hopper, David Molnar, and David Wagner. From weak to strong watermarking. In Salil P. Vadhan, editor, *TCC*, volume 4392 of *Lecture Notes in Computer Science*, pages 362–382. Springer, 2007.

[52] William A. Huber. GIS and steganography - part 3: Vector steganography, April 2002. `http://www.directionsmag.com/article.php?article_id=195`.

[53] Shingo Inoue, Kyoko Makino, Ichiro Murase, Osamu Takizawa, Tsutomu Matsumoto, and Hiroshi Nakagawa. A proposal on information hiding methods using XML. In *Worshop on Natural Language Processing and XML (beside NLPRS)*, page 55, 2001.

[54] Institut Géographique National. BD TOPO - descriptif technique (in french), december 2002. `http://www.ign.fr/telechargement/MPro/produit/BD_TOPO/JT_Agglo/DT_BDTOPOPays_1_2.pdf`.

[55] Paris C. Kanellakis, Gabriel M. Kuper, and Peter Z. Revesz. Constraint query languages. *Journal of Computer and System Sciences (JCSS)*, 51(1):26–52, August 1995.

[56] Hwan Il Kang, Kab Il Kim, and Jong Uk Cho. A vector watermarking using the generalized square mask. In *Proceedings of the International Conference on Information Technology: Coding and Computing*, pages 234–236, April 2001.

[57] Stefan Katzenbeisser and Fabien A. P. Petitcolas, editors. *Information hiding: techniques for steganography and digital watermarking*. Computer security series. Artech house, 2000.

[58] Stefan Katzenbeisser, Boris Škorić, Mehmet U. Celik, and Ahmad-Reza Sadeghi. Combining tardos fingerprinting codes and fingercasting. In *IH'07: Proceedings of the 9th international conference on Information hiding*, pages 294–310, Berlin, Heidelberg, 2007. Springer-Verlag.

[59] Auguste Kerckhoffs. La cryptographie militaire. *Journal des sciences militaires*, IX:5–38, January 1883.

[60] Auguste Kerckhoffs. La cryptographie militaire. *Journal des sciences militaires*, IX:161–191, February 1883.

[61] Sanjeev Khanna and Francis Zane. Watermarking maps: hiding information in structured data. In *Symposium on Discrete Algorithms (SODA)*, pages 596–605, 2000.

[62] P. Kolaitis and M. Thakur. Logical definability of NP optimization problems. *Information and Computation*, 115:321–353, 1994.

[63] Gabriel Kuper, Leonid Libkin, and Jan Paredaens, editors. *Constraint Databases*. Springer Verlag, 2000.

[64] Julien Lafaye and Jean Béguec. Watermarking plugin for OpenJump, 2007. `http://cedric.cnam.fr/~lafaye_j/index.php?n=Main.WaterGoatOpenJumpPlugin`.

[65] Julien Lafaye, Jean Béguec, David Gross-Amblard, and Anne Ruas. Invisible graffiti on your buildings: Blind and squaring-proof watermarking of geographical databases. In Dimitris Papadias, Donghui Zhang, and George Kollios, editors, *SSTD*, volume 4605 of *Lecture Notes in Computer Science*, pages 312–329. Springer, 2007.

[66] Julien Lafaye and David Gross-Amblard. XML streams watermarking. In *IFIP WG 11.3 Working Conference on Data and Applications Security (DBSEC)*, 2006.

[67] Julien Lafaye, David Gross-Amblard, Camélia Constantin, and Meryem Guerrouani. Watermill: An optimized fingerprinting system for databases under constraints. *IEEE Trans. Knowl. Data Eng. (TKDE)*, 20(4):532–546, 2008.

[68] Aurelien Lemay, Sebastian Maneth, and Joachim Niehren. A learning algorithm for top-down xml transformations. In *PODS '10: Proceedings of the twenty-ninth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems of data*, pages 285–296, New York, NY, USA, 2010. ACM.

[69] V.I. Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady*, 10(7):707–710, 1966.

[70] Qiming Li and Nasir D. Memon. Security models of digital watermarking. In Nicu Sebe, Yuncai Liu, Yueting Zhuang, and Thomas S. Huang, editors, *MCAM*, volume 4577 of *Lecture Notes in Computer Science*, pages 60–64. Springer, 2007.

[71] Yingjiu Li, Vipin Swarup, and Sushil Jajodia. Constructing a virtual primary key for fingerprinting relational data. In *Proceedings of the 2003 ACM workshop on Digital rights management*, pages 133–141. ACM Press, 2003.

[72] Yingjiu Li, Vipin Swarup, and Sushil Jajodia. A robust watermarking scheme for relational data. In *Proc. 13th Workshop on Information Technology and Systems (WITS 2003)*, pages 195–200, December 2003.

[73] Yingjiu Li, Vipin Swarup, and Sushil Jajodia. Fingerprinting relational databases: Schemes and specialties. *IEEE Trans. Dependable Sec. Comput. (TDSC))*, 2(1):34–45, 2005.

[74] Leonid Libkin and Limsoon Wong. Query languages for bags and aggregate functions. *Journal of Computer and System Sciences (JCSS)*, 55(2):241–272, 1997.

[75] Carlos Manuel Lopez Vazquez. Method of inserting hidden data into digital archives comprising polygons and detection methods, November 2003. US Patent no. 20030208679.

[76] Bertram Ludäscher, Pratik Mukhopadhyay, and Yannis Papakonstantinou. A transducer-based xml query processor. In *VLDB*, pages 227–238, 2002.

[77] Davide Martinenghi. Simplification of integrity constraints with aggregates and arithmetic built-ins. In *Flexible Query Answering Systems (FQAS)*, pages 348–361, 2004.

[78] Tova Milo, Dan Suciu, and Victor Vianu. Typechecking for XML transformers. *Journal of Computer and System Sciences (JCSS)*, 66(1):66–97, 2003.

[79] M. Monsignori, Paolo Nesi, and Marius B. Spinu. Watermarking music sheets. In *PCM '01: Proceedings of the Second IEEE Pacific Rim Conference on Multimedia*, pages 646–653, London, UK, 2001. Springer-Verlag.

[80] Alan Nash, Luc Segoufin, and Victor Vianu. Views and queries: Determinacy and rewriting. *ACM Trans. Database Syst.*, 35(3), 2010.

[81] F. Neven and T. Schwentick. Query automata on finite trees. *Theoretical Computer Science (TCS)*, 275:633–674, 2002.

[82] Wilfred Ng and Ho-Lam Lau. Effectives approaches for watermarking XML data. In *Proceedings of the 10th International Conference on Database Systems for Advanced Applications, DASFAA'05*, volume 3453 of *Lecture Notes in Computer Science*, pages 68–80, 2005.

[83] XiaMu Niu, ChengYong Shao, and XiaoTong Wang. A survey of digital vector map watermarking. *International Journal of Innovative Computing, Information and Control*, 2(6):1301–1316, December 2006.

[84] R. Ohbuchi, R.Ueda, and S. Endoh. Robust watermarking of vector digital maps. In *Proceedings of the International Conference on Multimedia and Expo, ICME '02*, volume 1, pages 577– 580, 2002.

[85] Ryutarou Ohbuchi, Hiroo Ueda, and Shuh Endoh. Watermarking 2D vector maps in the mesh-spectral domain. In *Proceedings of Shape Modeling International, SMI 2003*, pages 216–228, 2003.

[86] Christos H. Papadimitriou and Mihalis Yannakakis. Optimization, approximation, and complexity classes. *Journal of Computer and System Sciences (JCSS)*, 43(3):425–440, 1991.

[87] Kyi Tae Park, Kab Il Kim, Hwan Il Kang, and Seung-Soo Han. Digital geographical map watermarking using polyline interpolation. In *PCM '02: Proceedings of the Third IEEE Pacific Rim Conference on Multimedia*, pages 58–65, London, UK, 2002. Springer-Verlag.

[88] G. Qu and M. Potkonjak. *Intellectual Property Protection in VLSI Design: Theory and Practice*. Kluwer Publishing, February 2003.

[89] Gang Qu and Miodrag Potkonjak. Hiding signatures in graph coloring solutions. In *Information Hiding (IH)*, pages 348–367, 1999.

[90] Mikhail Atallah Radu Sion and Sunil Prabhakar. Resilient information hiding for abstract semi-structures. In Springer Verlag, editor, *Proceedings of the 4th Workshop on Digital Watermarking, IWDW'03*, volume 2939, pages 141–153, 2003.

[91] Laurent Raynal. Some elements for modelling updates in topographic databases. In *Proceedings of GIS/ LIS'96, Annual Conference and Exposition*, pages 1223–1232, Denver, Colorado, USA, 1996.

[92] Recordare. MusicXML Document Type Definition. `http://www.recordare.com/xml.html`.

[93] M. Sakamoto, Y. Matsuura, and Y. Takashima. A scheme of digital watermarking for geographical map data. In *Symposium on cryptography and information security*, Okinawa, Japan, January 2000.

[94] Craig Stuart Sapp. Online Database of Scores in the Humdrum File Format. In *Intl. Symp. on Music Information Retrievel*, pages 664–665, 2005.

[95] Hans Georg Schaathun. On watermarking/fingerprinting for copyright protection. In *ICICIC (3)*, pages 50–53. IEEE Computer Society, 2006.

[96] Martin Schmucker. Capacity improvement of a blind symbolic music score watermarking technique. In Wah Ping Wong, editor, *Security and Watermarking of Multimedia Contents IV.*, number 4675 in Proceedings of SPIE, pages 206–213, 2002.

[97] Martin Schmucker, Christoph Busch, and Anoop Pant. Digital watermarking for the protection of music scores. In Wah Ping Wong, editor, *Security and Watermarking of Multimedia Contents III*, number 4314 in Proceedings of SPIE, pages 85–95, Washington, 2001.

[98] Martin Schmucker and Hongning Yan. Music Score Watermarking by Clef Modifications. In Edward J. Delp, editor, *Security and Watermarking of Multimedia Contents V., SPIE Proceedings*, number 5020, pages 403–412, Bellingham, 2003.

[99] A. Schrijver. *Theory of linear and integer programming*. Wiley, 1986.

[100] Gerrit Schulz and Michael Voigt. A high capacity watermarking system for digital maps. In *Proceedings of the 2004 workshop on Multimedia and security*, pages 180–186, New York, NY, USA, 2004. ACM Press.

[101] Luc Segoufin and Victor Vianu. Validating streaming XML documents. In *Proceedings of 21st Symposium on Principles of Database Systems, PODS'02*, pages 53–64, 2002.

[102] Mohamed Shehab, Elisa Bertino, and Arif Ghafoor. Watermarking relational databases using optimization-based techniques. *IEEE Trans. Knowl. Data Eng. (TKDE)*, 20(1):116–129, 2008.

[103] Radu Sion, Mikhail Atallah, and Sunil Prabhakar. Rights protection for relational data. In *International Conference on Management of Data (SIGMOD)*, 2003.

[104] Radu Sion, Mikhail Atallah, and Sunil Prabhakar. Protecting rights over relational data using watermarking. *IEEE Trans. Knowl. Data Eng. (TKDE)*, 16(12):1509–1525, December 2004.

[105] Radu Sion, Mikhail J. Atallah, and Sunil Prabhakar. Resilient rights protection for sensor streams. In Mario A. Nascimento, M. Tamer Özsu, Donald Kossmann, Renée J. Miller, José A. Blakeley, and K. Bernhard Schiefer, editors, *VLDB*, pages 732–743. Morgan Kaufmann, 2004.

[106] Radu Sion, Mikhail J. Atallah, and Sunil Prabhakar. Rights protection for discrete numeric streams. *IEEE Trans. Knowl. Data Eng. (TKDE)*, 18(5):699–714, 2006.

[107] Larry J. Stockmeyer and Albert R. Meyer. Word problems requiring exponential time: Preliminary report. In *STOC*, pages 1–9. ACM, 1973.

[108] Swiss Society of Cartography. vol. 17: Topographic maps – map graphic and generalization, 2005.

[109] Gàbor Tardos. Optimal probabilistic fingerprint codes. In *Symposium on Theory of Computing (STOC)*, 2003.

[110] Gábor Tardos. Optimal probabilistic fingerprint codes. *J. ACM*, 55(2), 2008.

[111] The International Hydrographic Organization (IHO). *Specifications for Chart Content and Display Aspects of ECDIS*. IHO, 5th edition, december 2001.

[112] L. Valiant. The complexity of enumeration and reliability problems. *SIAM Journal of Computing*, 8(3), 1979.

[113] L. G. Valiant. A theory of the learnable. In *Symposium on Theory of Computing (STOC)*, pages 436–445, 1984.

[114] Michael Voigt and Christoph Busch. Watermarking 2D-vector data for geographical information systems. In Edward J. Delp and Ping Wah Wong, editors, *Proceedings SPIE-IS&T Electronic Imaging, SPIE*, volume 4675, pages 621–628, 2002.

[115] Michael Voigt and Christoph Busch. Feature-based watermarking of 2D vector data. In Edward J. Delp and Ping Wah Wong, editors, *Proceedings SPIE-IS&T Electronic Imaging, SPIE*, volume 5020, pages 359–366, June 2003.

[116] Gregory Wolfe, Jennifer L. Wong, and Miodrag Potkonjak. Watermarking graph partitioning solutions. In *DAC*, pages 486–489, 2001.

[117] Laurence A. Wolsey. *Integer Programming*. Wiley-interscience, 1998.

[118] Jennifer L. Wong, Gang Qu, and Miodrag Potkonjak. Optimization-intensive watermarking techniques for decision problems. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 23(1):119–127, 2004.

[119] Yuichiro Yonebayashi, Hirokazu Kameoka, and Shigeki Sagayama. Automatic decision of piano fingering based on a hidden markov models. In Manuela M. Veloso, editor, *IJCAI*, pages 2915–2921, 2007.

[120] Wenjun Zeng, Heather Yu, and Ching-Yung Lin, editors. *Multimedia security technologies for digital rights management*. Academic Press, July 2006.

[121] Xuan Zhou, HweeHwa Pang, and Kian-Lee Tan. Query-based watermarking for XML data. In *Proceedings of the ACM Symposium on Information, Computer and Communications Security, ASIACCS'07*, pages 253–264, March 2007.

[122] Marius Zimand. Weighted NP optimization problems: logical definability and approximation properties. *SIAM Journal of Computing*, 28(1):36–56, 1998.

Abstract

Database watermarking techniques allow for hiding information in a database, like a copyright mark. While watermarking methods are numerous in the multimedia setting, databases present various specificities. This work addresses some of them: how to watermark a numerical database while preserving the result of interesting aggregate queries, how to watermark a structured stream like a typed XML stream or a symbolic music score, how to watermark geographical data sets.

Résumé

Les techniques de tatouage de bases de données permettent la dissimulation d'information pertinente dans les n-uplets, comme par exemple l'identité du propriétaire des données. Les techniques de tatouage sont nombreuses dans le domaine multimédia, mais le tatouage des bases de données présente de nombreuses spécificités. Certaines d'entre elles sont traitées dans ce document : comment tatouer une base de données numérique tout en préservant le résultat de requêtes d'agrégat importantes, comment tatouer un flux structuré, comme un flux XML typé ou une partition musicale symbolique, comment tatouer une base de données géographiques.