Database Watermarking

Radu Sion

Network Security and Applied Cryptography Lab
Computer Science, Stony Brook University
sion@cs.stonybrook.edu

Abstract. As increasing amounts of data are produced, packaged and delivered in digital form, in a fast, networked environment, one of its main features threatens to become its worst enemy: zero-cost verbatim copies. The ability to produce duplicates of digital Works at almost no cost can now be misused for illicit profit. This mandates mechanisms for effective rights assessment and protection.

One such mechanism is based on *Information Hiding*. By concealing a resilient rights holder identity "signature" (*watermark*) within the digital Work(s) to be protected, Information Hiding for Rights Assessment (*Watermarking*) enables ulterior court-time proofs associating particular Works with their respective rights holders.

One main challenge is the fact that altering the Work in the process of hiding information could possibly destroy its value. At the same time one has to be concerned with a malicious adversary, with major incentives to remove or alter the watermark beyond detection – thus disabling the ability for court-time proofs – without destroying the value of the Work – to preserve its potential for illicit profit.

In this chapter we explore how Information Hiding can be deployed as an effective tool for Rights Assessment for discrete digital data. More specifically, we discuss numeric and categorical relational data.

1 Introduction

Mechanisms for privacy assurances (e.g., queries over encrypted data) are essential to a viable and *secure* management solution for outsourced data. On a somewhat orthogonal dimension but equally important, we find the requirement to be able to *assert and protect rights* over such data.

Different avenues are available, each with its advantages and drawbacks. Enforcement by legal means is usually ineffective, unless augmented by a digital counterpart such as Information Hiding. *Digital Watermarking* as a method of Rights Assessment deploys Information Hiding to conceal an indelible "rights witness" ("rights signature", watermark) within the digital Work to be protected (see Figure 1). The soundness of such a method relies on the assumption that altering the Work in the process of hiding the mark does not destroy the value of the Work, while it is difficult for a malicious adversary ("Mallory") to remove or alter the mark beyond detection without doing so. The ability to resist

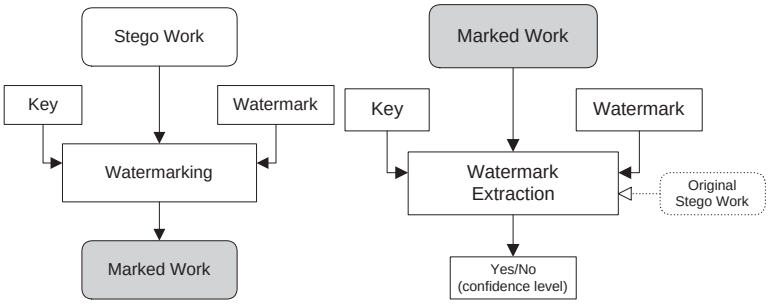

Fig. 1. Introduction: (a) *Digital Watermarking* conceals an indelible "rights witness" ("rights signature", watermark) within the digital Work to be protected. (b) In court, a detection process is deployed to prove the existence of this "witness" beyond reasonable doubt (confidence level) and thus assess ownership.

attacks from such an adversary, mostly aimed at removing the watermark, is one of the major concerns in the design of a sound solution.

There exists a multitude of semantic frameworks for discrete information processing and distribution. Each distinct data domain would benefit from the availability of a suitable watermarking solution.

Significant research efforts [2] [3] [8] [11] [14] [15] [22] [24] have been invested in the frameworks of signal processing and multimedia Works (e.g., images, video and audio).

Here we explore Information Hiding as a rights assessment tool for *discrete* data types i.e., in a relational database context. We explore existing watermarking solutions for numeric and categorical data types.

The Chapter is organized as follows. In Section 2 we explore the broader issues and challenges pertaining to steganography for rights protection. Then, in Sections 3 and 4 solutions for numeric respectively categorical data types are introduced. Related work is discussed in Section 5. Section 6 briefly discusses the current state of the art and Section 7 concludes.

2 Model

Before we proceed however, let us first understand how the ability to prove rights in court relates to the final desiderata, namely to *protect* those rights. After all, doesn't simply publishing a summary or digest of the Work to be protected – e.g., in a newspaper, just before releasing the Work – do the job? It would seem it enables one to prove later in court that (at least a copy of) the Work was in one's possession at the time of release. In the following we address these and other related issues.

2.1 Rights Protection through Assessment

The ability to prove/assess rights convincingly in court constitutes a deterrent to malicious Mallory. It thus becomes a tool for rights protection if counter-incentives and legal consequences are set high enough. But because Information Hiding does not provide a means of actual access control, the question of rights protection still remains. *How* are rights protected here?

It is intuitive that such a method works only if the rightful rights-holder (Alice) actually knows about Mallory's misbehavior **and** is able to prove to the court that: (i) Mallory possesses a certain Work X and (ii) X contains a "convincing" (e.g., very rare with respect to the space of all considered similar Works) and "relevant" watermark (e.g., the string "(c) by Alice").

What watermarking itself does not offer is a direct deterrent. If Alice does not have knowledge of Mallory's illicit possession of the Work and/or if it is impossible to actually prove this possession in court beyond reasonable doubt, then watermarking cannot be deployed directly to prevent Mallory. If, however, Information Hiding is aided by additional access control levels, it can become very effective.

For example, if in order to derive value from the given Work (e.g., watch a video tape), Mallory has to deploy a known mechanism (e.g., use video player), Information Hiding could be deployed to enable such a proof of possession, as follows: modify the video player so as to detect the existence of a watermark and match it with a set of purchased credentials and/or "viewing tickets" associated with the player's owner. If no match is found, the tape is simply not played back.

This scenario shows how watermarking can be deployed in conjunction with other technologies to aid in managing and protecting digital rights. Intuitively, a certain cost model is assumed here: the cost of reverse engineering this process is far higher than the potential derived illicit gain.

This illustrates the game theoretic nature at the heart of the watermarking proposition and of information security in general. Watermarking is a game with two adversaries, Mallory and Alice. At stake lies the value inherent in a certain Work X, over which Alice owns certain rights. When Alice releases X, to the public or to a licensed but potentially un-trusted party, she deploys watermarking for the purpose of ensuring that one of the following holds:

- she can always prove rights in court over any copy or valuable derivate of X (e.g., segment thereof)
- any existing derivate Y of X, for which she cannot prove rights, does not preserve any significant value (derived from the value in X)
- the cost to produce such an un-watermarked derivate Y of X that is still valuable (with respect to X) is higher than its value

Newspaper Digests To achieve the above however, Alice could publish a summary or digest (e.g., cryptographic hash) of X in a newspaper, thus being able to claim later on at least a time-stamp on the possession of X. This could apparently result in a quite effective, albeit costly, alternative to Watermarking the Work X.

There are many simple reasons why it would not work, including (i) scalability issues associated with the need for a trusted third party (newspaper), (ii) the cost of publishing a digest for each released Work, (iii) scenarios when the fact that the Work is watermarked should be kept secret (stealthiness) etc.

Maybe the most important reason however, is that Mallory can now claim that his ownership of the Work precedes X's publication date, and that Alice simply modified it (i.e., a stolen copy) and published a digest thereof herself. It would then be up to the court to decide if Mallory is to be believed or not, hardly an encouraging scenario for Alice. This could work if there existed a mechanism for the mandatory publication of digests for each and every valuable Work, again quite likely impractical due to both costs and lack of scalability to a virtually infinite set of data producers and Works.

Deploying such aids as rights assessment tools makes sense only in the case of the Work being of value only un-modified. In other words if it does not tolerate any changes, without losing its value, and Mallory is caught in possession of an identical copy, Alice can successfully prove in court that she possessed the original at the time of its publication (but she cannot prove more). Considering that, in the case of watermarking, the assumption is that, no matter how small, there are modifications allowed to the Works to be protected, in some sense the two approaches complement each other. If no modifications are allowed, then a third-party "newspaper" service might work for providing a time-stamp type of ownership proof that can be used in court.

Steganography and Watermarking There exists a fundamental difference between Watermarking and generic Information Hiding (steganography) from an application perspective and associated challenges. Information Hiding in general (and covert communication in particular), aims usually at enabling Alice and Bob to exchange messages in a manner as resilient and stealthy as possible, through a hostile medium where Malory could lurk. On the other hand, Digital Watermarking is deployed by Alice as a court proof of rights over a Work, usually in the case when Mallory benefits from using/selling that very same Work or maliciously modified versions of it.

In Digital Watermarking, the actual value to be protected lies in the Works themselves whereas pure steganography usually makes use of them as simple value "transporters". In Watermarking, Rights Assessment is achieved by demonstrating (with the aid of a "secret" known only to Alice – "watermarking key") that a particular Work exhibits a rare property ("hidden message" or "watermark"). For purposes of convincing the court, this property needs to be so *rare* that if one considers any other random Work "similar enough" to the one in question, this property is "very improbable" to apply (i.e., bound false-positives rate). It also has to be *relevant*, in that it somehow ties to Alice (e.g., by featuring the bit string "(c) by Alice").

There is a threshold determining the ability to convince the court, related to the "very improbable" assessment. This defines a main difference from steganography: from the court's perspective, specifics of the property (e.g., watermark

message) are not important as long as they link to Alice (e.g., by saying "(c) by Alice") and, she can prove "convincingly" it is she who induced it to the (non-watermarked) original.

In watermarking the emphasis is on "detection" rather than "extraction". Extraction of a watermark, or bits of it, is usually a part of the detection process but just complements the process up to the extent of increasing the ability to convince in court. If recovering the watermark data in itself becomes more important than detecting the actual existence of it (i.e., "yes/no" answer) then, from an application point of view, this is a drift toward covert communication and pure Information Hiding (steganography).

2.2 Consumer Driven Watermarking

An important point about watermarking should be noted. By its very nature, a watermark modifies the item being watermarked: it inserts an indelible mark in the Work such that (i) the insertion of the mark does not destroy the value of the Work, i.e., it is still useful for the *intended purpose*; and (ii) it is difficult for an adversary to remove or alter the mark beyond detection without destroying this value. If the Work to be watermarked cannot be modified without losing its value then a watermark cannot be inserted. The critical issue is not to avoid alterations, but to limit them to acceptable levels with respect to the intended use of the Work.

Thus, an important first step in inserting a watermark, i.e., by altering it, is to identify changes that are acceptable. Naturally, the nature and level of such change is dependent upon the application for which the data is to be used. Clearly, the notion of value or utility of the data becomes thus central to the watermarking process. For example, in the case of software, the value may be in ensuring equivalent computation, whereas for natural language text it may be in conveying the same meaning – i.e., synonym substitution is acceptable. Similarly, for a collection of numbers, the utility of the data may lie in the actual values, in the relative values of the numbers, or in the distribution (e.g., normal with a certain mean). At the same time, the concept of value of watermarked Works is necessarily relative and largely influenced by each semantic context it appears in. For example, while a statistical analyst would be satisfied with a set of feature summarizations (e.g., average, higher-level moments) of a numeric data set, a data mining application may need a majority of the data items, for example to validate a classification hypothesis.

It is often hard to define the available "bandwidth" for inserting the watermark directly. Instead, allowable distortion bounds for the input data can be defined in terms of consumer metrics. If the watermarked data satisfies the metrics, then the alterations induced by the insertion of the watermark are considered to be acceptable. One such simple yet relevant example for numeric data, is the case of *maximum allowable mean squared error* (MSE), in which the usability metrics are defined in terms of mean squared error tolerances as $(s_i - v_i)^2 < t_i, \forall i = 1, ..., n$ and $\sum (s_i - v_i)^2 < t_{max}$, where $\mathbb{S} = \{s_1, ..., s_n\} \subset \mathbb{R}$, is the data to be watermarked, $\mathbb{V} = \{v_1, ..., v_n\}$ is the result, $\mathbb{T} = \{t_1, ..., t_n\} \subset \mathbb{R}$

and $t_{max} \in \mathbb{R}$ define the guaranteed error bounds at data distribution time. In other words $\mathbb{T}$ defines the allowable distortions for individual elements in terms of MSE and t_{max} its overall permissible value.

Often however, specifying only allowable change limits on individual values, and possibly an overall limit, fails to capture important semantic features associated with the data – especially if the data is structured. Consider for example, age data. While a small change to the age values may be acceptable, it may be critical that individuals that are younger than 21 remain so even after watermarking if the data will be used to determine behavior patterns for under-age drinking. Similarly, if the same data were to be used for identifying legal voters, the cut-off would be 18 years. Further still, for some other application it may be important that the relative ages, in terms of which one is younger, not change. Other examples of constraints include: (i) *uniqueness* – each value must be unique; (ii) *scale* – the ratio between any two number before and after the change must remain the same; and (iii) *classification* – the objects must remain in the same class (defined by a range of values) before and after the watermarking. As is clear from the above examples, simple bounds on the change of numerical values are often not enough.

Structured collections, present further constraints that must be adhered to by the watermarking algorithm. Consider a data warehouse organized using a standard Star schema with a fact table and several dimension tables. It is important that the key relationships be preserved by the watermarking algorithm. This is similar to the "Cascade on update" option for foreign keys in SQL and ensures that tuples that join before watermarking also join after watermarking. This requires that the new value for any attribute should be unique after the watermarking process. In other words, we want to preserve the relationship between the various tables. More generally, the relationship could be expressed in terms of an arbitrary join condition, not just a natural join. In addition to relationships between tuples, relational data may have constraints within tuples. For example, if a relation contains the start and end times of a web interaction, it is important that each tuple satisfies the condition that the end time be later than the start time.

There exists a trade-off between the desired level of marking resilience and resistance to attacks, and the ability to preserve data quality in the result, with respect to the original. Intuitively, at the one extreme, if the encoded watermark is to be very "strong" one can simply modify the *entire* data set aggressively, but at the same time probably also destroy its actual value. As data quality requirements become increasingly restrictive, any applied watermark is necessarily more vulnerable. Often we can express the available bandwidth as an increasing function of allowed alterations. At the other extreme, a disproportionate concern with data quality will hinder most of the watermarking alterations, resulting in a weak, possibly non-existent encoding.

Naturally, one can always identify some use that is affected by even a minor change to any portion of the data. It is therefore important that (i) the main intended purpose and semantics that should be preserved be identified

during watermarking and that (ii) *the watermarking process not interfere with the final data consumer requirements*. We call this paradigm *consumer driven watermarking*.

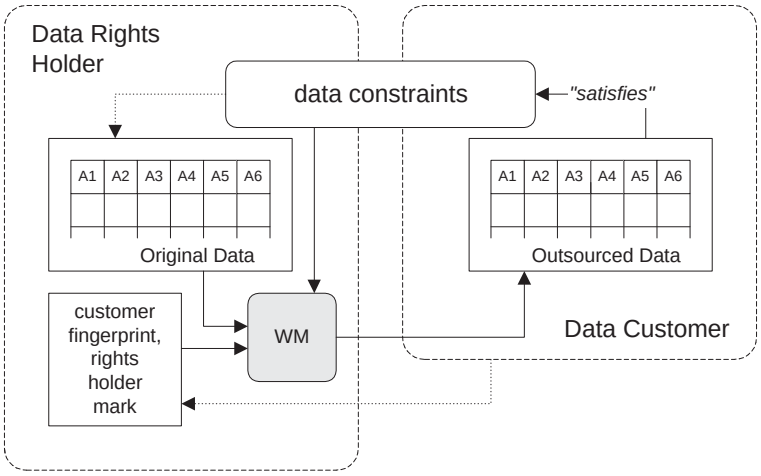

Fig. 2. In *consumer-driven watermarking* a set of data constraints are continuously evaluated in the encoding process to ensure quality of the result.

Some of the solutions discussed here are consumer driven enabled through feedback mechanisms (see Figure 2) that allow the watermarking process to "rollback" modifications that would violate quality constraints in the result on a step by step basis. This ensures the preservation of desired quality metrics with respect to the original un-watermarked input Work.

2.3 Discrete Data vs. Multimedia

An established body of research [2] [3] [8] [11] [14] [15] [22] [24] has resulted from work on Information Hiding and Watermarking in frameworks such as signal processing and multimedia (e.g., images, video and audio). Here we explore Information Hiding as a rights assessment tool for *discrete* data types.

Let us briefly explore the relationship between the challenges and techniques deployed in both frameworks. Because, while the terms might be identical, the associated models, challenges and techniques are different, almost orthogonal. Whereas in the signal processing case there usually exists a large noise bandwidth, due to the fact that the final data consumer is likely human – with associated limitations of the sensory system – in the case of discrete data types this cannot be assumed and data quality assessment needs to be closely tied with the actual watermarking process (see Section 2.2).

Another important differentiating focus is the emphasis on the actual ability to convince in court as a success metric, unlike most approaches in the signal processing realm, centered on bandwidth. While bandwidth is a relevant related metric, it does not consider important additional issues such as malicious transforms and removal attacks. For rights assertion, the concerns lie not as much with packing a large *amount* of information (i.e., watermark bits) in the Works to be protected, as with being able to both *survive* removal attacks and *convince* in court.

Maybe the most important difference between the two domains is that, while in a majority of watermarking solutions in the multimedia framework, the main domain transforms are signal processing primitives (e.g., Works are mainly considered as being compositions of signals rather than strings of bits), in our case data types are mostly discrete and are not naturally handled as continuous signals. Because, while discrete versions of frequency transforms can be deployed as primitives in information encoding for digital images [8], the basis for doing so is the fact that, although digitized, images are at the core defined by a composition of light reflection signals and are consumed as such by the final human consumer. By contrast, arbitrary discrete data is naturally discrete [1] and often to be ingested by a highly sensitive semantic processing component, e.g., a computer rather than a perceptual system tolerant of distortions.

2.4 Relational Data

For completeness let us briefly overview main components of a relational model [7]. In such a model, relations between information items are explicitly specified: data is organized as "a number of differently sized *tables*" [7] composed of "related" rows/columns. A table is a collection of *rows* or records and each row in a table contains the same *fields*. Certain fields may be designated as data *keys* (not to be confused with "cryptographic keys") when a functional dependency or key constraint, holds for the table. Often, indexing is deployed to speed up searches on values of such primary key fields. Data is structured logically into valued *attributes*. From this perspective, a table is a collection of such attributes (the columns of the table) and models a *relation* among them. The data rows in the tables are also called *tuples*. Data in this model is manipulated using a *relational algebra*. Main operations in this algebra are set operations (e.g., union, intersection, Cartesian product), selection (of some tuples in tables) and projection (of some columns/attributes).

Rights protection for such data is important in scenarios where it is sensitive, valuable and about to be outsourced. A good example is a data mining application, where data is sold in pieces to parties specialized in mining it, e.g., sales patterns database, oil drilling data, financial data. Other scenarios involve for example online B2B interactions, e.g., airline reservation and scheduling portals, in which data is made available for direct, interactive use (see Figure 3). Given

[1] Unless we consider quantum states and uncertainty arising in the spin of the electrons flowing through the silicon.

the nature of most of the data, it is hard to associate rights of the originator over it. Watermarking can be used to solve this issue.

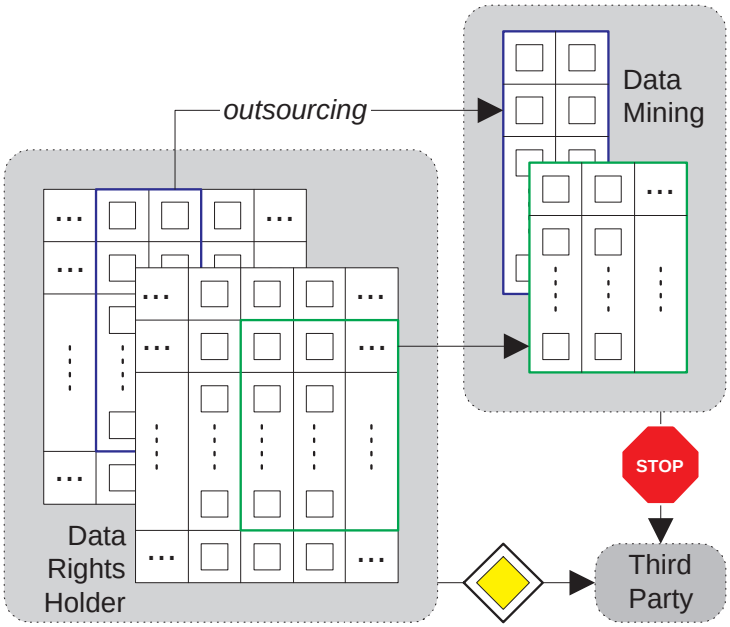

Fig. 3. Rights Assessment is important when valuable data is outsourced to a third party.

2.5 The Adversary

Watermarking is a game between the watermarker and malicious Mallory. In this game, the watermarker and Mallory play against each other within subtle trade-off rules aimed at keeping the quality of the result within acceptable bounds. It is as if there exists an impartial referee (the data itself) moderating each and every "move". As discussed above, it is important to make this "referee" an explicit part of the marking process (consumer-driven paradigm). It is also important to understand Mallory and the adversarial setting.

Once outsourced, i.e., out of the control of the watermarker, data might be subjected to a set of attacks or transformations; these may be malicious – e.g., with the explicit intent of removing the watermark – or simply the result of normal use of the data. An effective watermarking technique must be able to survive

such use. In a relational data framework important attacks and transformations are:

A1. Sampling. The attacker (Mallory) can randomly select and use a subset of the watermarked data set that might still provide value for its intended purpose ("subset selection"). More specifically, here we are concerned with both (**A1.a**) horizontal and (**A1.b**) vertical data partitioning – in which a valuable subset of the *attributes* are selected by Mallory.

A2. Data Addition. Mallory adds a set of tuples to the watermarked set. This addition is not to significantly alter the useful properties of interest to Mallory.

A3. Alteration. Altering a subset of the items in the watermarked data set such that there is still value associated with the result. In the case of numeric data types, a special case needs to be outlined here, namely (**A3.a**) a linear transformation performed uniformly to all of the items. This is of particular interest as it can preserve significant valuable data-mining related properties of the data.

A4. Ulterior Claims of Rights. Mallory encodes an additional watermark in the already watermarked data set and claims rights based upon this second watermark.

A5. Invertibility Attack. Mallory attempts to establish a plausible (watermark,key) pair that matches the data set and then claims rights based on this found watermark [8, 9].

Given the attacks above, several properties of a successful solution surface. For immunity against **A1**, the watermark has to be likely encoded in overall data properties that survive sampling, e.g., confidence intervals, statistical bias. With respect to (**A1.b**) special care has to be taken such that the mark survives this partitioning. The encoding method has to feature a certain attribute-level property that could be recovered in such a vertical partition of the data. We believe that while vertical data partitioning attacks are possible and also very likely in certain scenarios, often value is to be found in the association between a set of relation attributes. These attributes are highly likely to survive such an attack, as the final goal of the attacker is to produce a still-valuable result. If the assumption is made that the attack alterations do not destroy the value of the data, then **A3** could be handled by encoding the primitive mark in resilient global data properties. As a special case, **A3.a** can be resisted by a preliminary normalization step in which a common divider to all the items is first identified and applied.

While powerful, for arbitrary watermarks, the invertibility attack **A5** can be defeated by requiring the encoded string to be relevant (e.g. "(c) by Mallory") and the encoding to be "convincing" (see Section 2.1). Then the probability of success of invertibility searches becomes upper bound.

In order to defeat **A4**, the watermarking method has to provide the ability to determine encoding precedence, e.g., if it can be proved in court that one watermark encoding was "overwritten" by a later one. Additionally, in the case of such a (court time) dispute, the parties could be requested to present a portion

of the original, un-watermarked data. Only the rightful rights holder would be able to produce such a proof, as Mallory could only have access to already watermarked data.

It is worth also noting that, intuitively, if, in the process of watermarking, the data is altered to its usability limits, any further alteration by a watermarker is likely bound to yield an unusable result. Achieving this might be often desirable [2] and has been explored by Sion et. al. in a proof of concept implementation [34] as well as by Li et. al. in [20] (this is discussed in more detail elsewhere in this book). The challenges of achieving such a desiderata however, lies in the impossibility to define absolute data quality metrics that consider all value dimensions of data.

3 Numeric Types

In this section we explore watermarking solutions in the context of relational data in which one or more of the attributes are of a numeric type. Among existing solutions we distinguish between *single-bit* (the watermark is composed of a single bit) and *multi-bit* (the watermark is a string of bits) types. Orthogonally, the encoding methods can be categorized into two; we chose to call them *direct-domain* and *distribution* encodings. In a direct-domain encoding, each individual bit alteration in the process of watermarking is directly correlated to (a part of) the encoded watermark. In distribution encodings, the encoding channel lies often in higher order moments of the data (e.g., running means, hierarchy of value averages). Each individual bit alteration impacts these moments for the purpose of watermark encoding, but in itself is not directly correlated to any one portion of the encoded watermark.

Single Bit Direct Domain Encoding In [1, 16] Kiernan, Agrawal et.al. propose a direct domain encoding of a single bit watermark in a numeric relational database.

Overview. Its main algorithm proceeds as follows. A subset of tuples are selected based on a secret criteria; for each tuple, a secret attribute and corresponding least significant (ξ) bit position are chosen. This bit position is then altered according to yet another secret criteria that is directly correlated to the watermark bit to be encoded. The main assumption is, that changes can be made to any attribute in a tuple at any least significant ξ bit positions. At watermark detection time, the process will re-discover the watermarked tuples and, for each detected accurate encoding, become more "confident" of a true-positive detection.

There are a set of important assumptions underlying this method. Maybe the most important one is that "the relational table being watermarked is such that if all or a large number of the ξ least significant bits of any attribute are dropped or perturbed, then the value of the data is significantly reduced. However, it is

[2] This is formulated as the "optimality principle" in [26], as well as previous results such as [28] and [31].

possible to change a small number of the bits and not decrease the value of the data significantly" [16].

The authors make an argument for this being a reasonable assumption as such techniques have been used by publishers of books of mathematical tables for a long time – e.g., by introducing small errors in published logarithm tables and astronomical ephemerides to identify pirated copies [15]. Examples of real-world data sets that satisfy such an assumption are given, including tables of parametric specifications (mechanical, electrical, electronic, chemical, etc.), surveys (geological, climatic, etc.), and life sciences data (e.g., gene expression).
Solution Details. For consistency, the original notation is used: a database relation R with the following schema is $R(P, A_0, \ldots, A_{\nu-1})$, is assumed, with P the primary key attribute. All ν attributes $A_0, \ldots, A_{\nu-1}$ are candidates for marking: the values are assumed such that small changes in the ξ least significant bits are imperceptible. γ denotes a control parameter that determines the average number ω of tuples marked ($\omega = \frac{\eta}{\gamma}$), where η is the number of tuples in the database. $r.X$ is used to denote the value of attribute X in tuple r, α denotes a "significance level" and τ a "threshold" for the test of "detecting a watermark". $\mathcal{K}$ is a key known only to the database owner, and there exists $\mathcal{G}$, a pseudo-random sequence number generator [23] (next($\mathcal{G}$) denotes the next generated sequence number).

Note: There are a set of changes between the initial proposed scheme in [16] and its journal version [1]. Here we discuss the (more robust) journal version.

---

1) **foreach** tuple $r \in R$ **do**
2) seed $\mathcal{G}$ with $r.P$ concatenated with $\mathcal{K}$
3) **if** (next($\mathcal{G}$) mod $\gamma = 0$) **then** // mark this tuple
4) attribute_index i = next($\mathcal{G}$) mod ν // mark attribute A_i
5) bit_index j = next($\mathcal{G}$) mod η // mark j^{th} bit
6) $r.A_i$ = mark(next($\mathcal{G}$),$r.A_i$,j)
7) mark(random_number i, value v, bit_index j) **return** value
8) **if** (i is even) **then**
9) set the j^{th} least significant bit of v to 0
10) **else**
11) set the j^{th} least significant bit of v to 1
12) **return** v

Fig. 4. Watermark insertion for the single-bit encoding of [1, 16].

Watermark insertion is illustrated in Figure 4. The main steps of the algorithm are as follows. Initially (step 2) the random sequence generator is initialized such that its output is distinct for any given distinct tuple value. This mechanism is deployed in order to achieve a certain tuple ordering independence of the encoding. The output of $\mathcal{G}$ is then used to determine: (i) if the current tuple is to

be watermarked (step 3), (ii) which attribute value to mark (step 4), (iii) which bit within that attribute's value to alter (step 5), and (iv) what new bit-value to assign to that bit-position in the result (step 6, invocation of mark()). This encoding guarantees that, in order to entirely remove a watermark, Mallory is put in the position of guessing correctly the marked tuples, attributes and altered bit positions.

Once R is published, the data owner, Alice, would like to determine whether the (similar) relation S published by Mallory has been pirated from R. The sets of tuples and of attributes in S are assumed to be strict subsets of those in R. Additionally, Mallory is assumed not to drop the primary key attribute or change the value of primary keys. Then watermark detection is a direct inverse of insertion. It proceeds as follows (see Figure 5).

---

1) totalcount = matchcount = 0
2) **foreach** tuple $s \in S$ **do**
3) seed $\mathcal{G}$ with $s.P$ concatenated with $\mathcal{K}$
4) **if** (next($\mathcal{G}$) mod $\gamma = 0$) **then** // tuple was marked
5) attribute_index i = next($\mathcal{G}$) mod ν // A_i was marked
6) bit_index j = next($\mathcal{G}$) mod η // j^{th} bit was marked
7) totalcount = totalcount + 1
8) matchcount = matchcount + match (next($\mathcal{G}$,$s.A_i$,j)
9) τ = threshold(totalcount,α)
10) **if** ((matchcount < τ) **or** (matchcount > totalcount - τ)) **then**
11) *suspect piracy*
12) match(random_number i, value v, bit_index j) **return** integer
13) **if** (i is even) **then**
14) **return** 1 **if** the j^{th} least significant bit of v is 0 **else return** 0
15) **else**
16) **return** 1 **if** the j^{th} least significant bit of v is 1 **else return** 0

Fig. 5. Watermark detection for the single-bit encoding of [1, 16].

Alice starts by identifying the bits that should have been marked by the insertion algorithm. To do so, it executes the operations described in lines 1 through 5 of the insertion algorithm (steps 3 through 6). The assumption is that the original database primary key is preserved in S. Each such identified bit is tested for a match with the value that should have been assigned by the insertion algorithm. Each match is counted. If the resulting count is either too small or too large, piracy is suspected. In the case of too small a number, the method assumes that somehow Mallory has identified the marked bits and systematically flipped each one.

In other words, the insertion algorithm is modulated on a set of successive independent coin tosses. A detection algorithm over ω bits will yield a number of matches with a binomial distribution $(\omega, 1/2)$ for the null hypothesis of non-

piracy. Naturally, in the absence of piracy, the expected number of matches is $\frac{\omega}{2}$. The paper proposes to suspect piracy if the observed number of matches m is so large or so small that its probability under the null hypothesis is highly unlikely.

This can be modeled by first fixing an acceptable value for the *significance level* $\alpha \in (0, 1)$ and then computing a threshold $\tau \in (0, \frac{\omega}{2})$ such that the probability of $m < \tau$ or $m > \omega - \tau$ under the null hypothesis is less than or equal to α.

The authors discuss additional extensions and properties of the solution including the following:

- Incremental Updatability: Updates can be handled independently of the existing watermark as the selection and marking criteria are self-sufficient and only depend on the primary key value.
- Blind Watermarking: The method does not require the availability of the un-watermarked data at detection time.
- Varying Parameters: The assumption that any two attributes are marked at the same rate can be removed. Different attributes can be marked at different rates because the attributes may tolerate different error rates and, if the rate parameters are secret, Mallory's task become even more difficult. Additionally, the number of bits available for marking can be varied from one attribute to another.
- Relations Without Primary Keys: The authors also discuss extensions aimed at handling the case of relations without primary keys. This is an important problem as it has the potential to overcome the required assumption of unchanged primary key values in the watermarked data at detection time. In the case of no primary key, the authors propose to designate another attribute, or a number of most significant bit-portions of the currently considered one, as a primary key. This however presents a significant vulnerability due to the very likely existence of duplicates in these values. Mallory could mount a statistical attack by correlating marked bit values among tuples with the same most significant bits. This issue has been also considered in [18] where a similar solution has been adopted. This, is discussed in more detail elsewhere in this book.

3.1 Multi-Bit Watermarks

While there likely exist applications whose requirements are satisfied by single-bit watermarks, often it is desirable to provide for "relevance", i.e., linking the encoding to the rights holder identity. This is especially important if the watermark aims to defeat against invertibility attacks (**A5**).

In a single-bit encoding this can not be easily achieved. Additionally, while the main proposition of watermarking is not covert communication but rather rights assessment, there could be scenarios where the actual message payload is of importance.

One apparent direct extension from single-bit watermarks to a multi-bit version would be to simply deploy a different encoding, with a separate watermark key, for each bit of the watermark to be embedded. This however, might not be possible, as it will raise significant issues of inter-encoding interference: the encoding of later bits will likely distort previous ones. This will also make it harder to handle ulterior claim of rights attacks (**A4**).

In the following we discuss multi-bit watermark encodings. We briefly discuss a direct-domain encoding [19] that extends the work by Kiernan, Agrawal et. al. [1, 16] and then explore a distribution-encoding method by Sion et. al. [27, 29, 30, 32, 33] and [34].

Multi-Bit Direct Domain Encoding In [19] Li et. al. extend the work by Kiernan, Agrawal et. al. [1, 16] to provide for multi-bit watermarks in a direct domain encoding. This is discussed in extended detail elsewhere in this book. Here we briefly summarize. The scheme functions as follows. The database is parsed and, at each bit-encoding step, one of the watermark bits is randomly chosen for embedding; the solution in [1, 16] is then deployed to encode the selected bit in the data at the "current" point. The "strength of the robustness" of the scheme is claimed to be increased with respect to [1, 16] due to the fact that the watermark now possesses an additional dimension, namely length. This should guarantee a better upper bound for the probability that a valid watermark is detected from unmarked data, as well as for the probability that a fictitious secret key is discovered from pirated data (i.e., invertibility attacks **A5**). This upper bound is said to be independent of the size of database relations thus yielding robustness against attacks that change the size of database relations.

Multi-Bit Distribution Encoding Encoding watermarking information in resilient numeric distribution properties of data presents a set of advantages over direct domain encoding, the most important one being its increased resilience to various types of numeric attacks. In [27, 29, 30, 32, 33] and [34], Sion et. al. introduce a multi-bit distribution encoding watermarking scheme for numeric types. The scheme was designed with both an adversary and a data consumer in mind. More specifically the main desiderata were: (i) watermarking should be consumer driven – i.e., desired semantic constraints on the data should be preserved – this is enforced by a feedback-driven rollback mechanism, and (ii) the encoding should survive important numeric attacks, such as linear transformation of the data (**A3.a**), sampling (**A1**) and random alterations (**A3**).
Overview. The solution starts by receiving as user input a reference to the relational data to be protected, a watermark to be encoded as a copyright proof, a secret key used to protect the encoding and a set of data quality constraints to be preserved in the result. It then proceeds to watermark the data while continuously assessing data quality, potentially backtracking and rolling back undesirable alterations that do not preserve data quality.

Watermark *encoding* is composed of two main parts: in the first stage, the input data set is securely partitioned into (secret) subsets of items; the second

stage then encodes one bit of the watermark into each subset. If more subsets (than watermark bits) are available, error correction is deployed to result in an increasingly resilient encoding. Each single bit is encoded/represented by introducing a slight skew bias in the tails of the numeric distribution of the corresponding subset. The encoding is proved to be resilient to important classes of attacks, including subset selection, linear data changes and random item(s) alterations.

Solution Details. The algorithm proceeds as follows (see Figure 6): **(a)** User-defined queries and associated guaranteed query usability metrics and bounds are specified with respect to the given database (see below). **(b)** User input determines a set of attributes in the database considered for watermarking, possibly all. **(c)** From the values in each such attribute select a (maximal) number of (e) unique, non-intersecting, secret subsets. **(d)** For each considered subset, **(d.1)** embed a watermark bit into it using the single-bit encoding convention described below and then **(d.2)** check if data constraints are still satisfied. If data constraints are violated, **(d.3)** retry different encoding parameter variations or, if still no success, **(d.4)** try to mark the subset as invalid (see single-bit encoding convention below), or if still no success **(d.5)** ignore the current set[3]. Repeat step (d) until no more subsets are available.

---

wm(attribute, wm_key, mark_data[],
plugin_handler, db_primary_key, subset_size, v_{false}, v_{true}, c)
sort_attribute ← **sort_on_normalized_hash**(wm_key,db_primary_key,wm_key)
for (i=0; i < $\frac{length(attribute)}{subset\_size}$;i++)
subset_bin ← **next** subset_size elements from sort_attribute
compute rollback_data
encode(mark_data[i % mark_data.length], subset_bin, v_{false}, v_{true}, c)
propagate changes into attribute
if (not goodness_plugin_handler.isSatisfied(new_data,changes)) **then**
rollback rollback_data
continue
else
commit
map[i] = true
subset_boundaries[i] = subset_bin[0]
return map, subset_boundaries

Fig. 6. Watermark Embedding (version using subset markers and detection maps shown).

[3] This leaves an invalid watermark bit encoded in the data that will be corrected by the deployed error correcting mechanisms (e.g. majority voting) at extraction time.

Several methods for subset selection (c) are discussed. In one version, it proceeds as follows. The input data tuples are sorted (lexicographically) on a secret keyed cryptographic hash H of the primary key attribute (K). Based on this value, compose a criteria (e.g., $H(K, key)$) mod $e = 0$) for selecting a set of "special" tuples such that they are uniformly distributed and average a total number of $e = length(attribute)/\ subset\_size$. These special tuples are going to be used as subset "markers". Each subset is defined as the elements between two adjacent markers, having on average $subset\_size$ elements. The detection phase will then rely on this construction criteria to re-discover the subset markers. This process is illustrated in Figure 6.

Encoding the individual mark bits in different subsets increases the ability to defeat different types of transformations including sampling (**A1**) and/or random data addition (**A2**), by "dispersing" their effect throughout the data, as a result of the secret ordering. Thus, if an attack removes 5% of the items, this will result in each subset S_i being roughly 5% smaller. If S_i is small enough and/or if the primitive watermarking method used to encode parts of the watermark (i.e., 1 bit) in S_i is made resilient to these kind of minor transformations then the probability of survival of most of the embedded watermarks is accordingly higher. Additionally, in order to provide resilience to massive "cut" attacks, the subsets are made to be of sizes equal to a given *percent* of the overall data set, i.e., not of fixed absolute sizes.

Note: If enough additional storage is available, these subsets can be in fact constructed differently: given a secretly keyed cryptographic hash function with discrete output values in the interval $[1, e]$, apply it, for each tuple, to the primary key attribute value and let its output determine which subset the tuple belongs to. This would both alleviate the need to deploy subset markers as well as likely offering more resilience to attacks. This simple and nice improvement was suggested to one of the authors during a discussion with a Purdue graduate student (whose identity he cannot remember but whom he invites forward for credit) attending the 2005 Symposium on Security and Privacy.

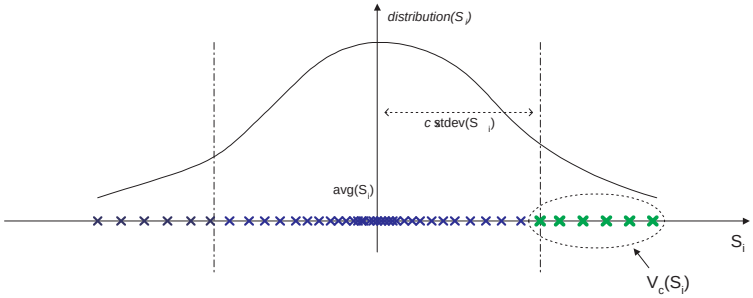

Fig. 7. In the single-bit mark encoding convention, the encoding of the watermark bit relies on altering the size of the "positive violators" set, $v_c(S_i)$.

Once constructed, each separate subset S_i will be marked separately with a single bit, in the order it appears in the actual watermark string. The result will be a e-bit (i.e., $i = 1, \ldots, e$) overall watermark bandwidth in which each bit is "hidden" in each of the marked S_i. If the watermark is of size less than e, error correction can be deployed to make use of the additional bandwidth to increase the encoding resilience.

The single-bit distribution encoding proceeds as follows. Let b be a watermark bit that is to be encoded into S_i and $\mathbb{G}$ represent a set of user specified change tolerance, or usability metrics. The set $\mathbb{G}$ will be used to implement the consumer-driven awareness in the watermark encoding.

Let v_{false}, v_{true}, $c \in (0, 1)$, $v_{false} < v_{true}$ be real numbers (e.g., $c = 90\%$, $v_{true} = 10\%$, $v_{false} = 7\%$). c is called *confidence factor* and the interval (v_{false}, v_{true}) *confidence violators hysteresis*. These are values to be remembered also for watermark detection time. They can be considered as part of the encoding key. Let $avg(S_i)$ and $\delta(S_i)$ be the average and standard deviation, respectively, of S_i. Given S_i and the real number $c \in (0, 1)$ as above, $v_c(S_i)$ is defined as the *number of items of* S_i *that are greater than* $avg(S_i) + c \times \delta(S_i)$. $v_c(S_i)$ is called the number of positive "violators" of the c confidence over S_i, see Figure 7.

The single-bit ***mark encoding convention*** is then formulated: given S_i, c, v_{false} and v_{true} as above, $mark(S_i) \in \{true, false, invalid\}$ is defined to be $true$ if $v_c(S_i) > (v_{true} \times |S_i|)$, $false$ if $v_c(S_i) < v_{false} \times |S_i|$ and $invalid$ if $v_c(S_i) \in (v_{false} \times |S_i|, v_{true} \times |S_i|)$.

In other words, the watermark is modeled by the percentage of positive confidence violators present in S_i for a given confidence factor c and confidence violators hysteresis (v_{false}, v_{true}). Encoding the single bit (see Figure 8), b, into S_i is therefore achieved by minor alterations to some of the data values in S_i such that the number of positive violators $(v_c(S_i))$ is either (a) less than $v_{false} \times |S_i|$ if $b = 0$, or (b) more than $v_{true} \times |S_i|$ if $b = 1$. The alterations are then checked against the change tolerances, $\mathbb{G}$, specified by the user.

At detection time the secret subsets are reconstructed and the individual bits are recovered according to the single-bit mark encoding convention. This yields the original e-bit string. If e is larger than the size of the watermark, error correction was deployed to increase the encoding resilience. The watermark string can be then recovered by applying error correction decoding to this string, e.g., majority voting for each watermark bit. This process is illustrated in Figure 9.

In [27,33] and [34] the authors discuss a proof of concept implementation. It is worth mentioning here due to its consumer-driven design (see Figure 10). In addition to a watermark to be embedded, a secret key to be used for embedding, and a set of relations/attributes to watermark, the software receives as input also a set of external *usability plugin modules*. The role of these plugins is to allow user defined query metrics to be deployed and queried at run-time without recompilation and/or software restart. The software uses those metrics to re-evaluate data usability after each atomic watermarking step.

```
encode(bit, set, v_false, v_true, c)
    compute avg(set), δ(set)
    compute v_c(set)
    if v_c(set) satisfies desired bit value return true
    if (bit)
        compute v_* ← v_true − v_c(set)
        alter v_* items close to the stddev boundary so that they become > v_true
    else
        (!bit) case is similar
    compute v_c(set)
    if v_c(set) satisfies desired bit value return true
    else rollback alterations (distribution shifted too much?)
    return false
```

Fig. 8. Single Bit Encoding Algorithm (illustrative overview).

Constraint metrics can be specified either as SQL queries, stored procedures or simple Java code inside the plug-in modules. Constraints that arise from the schema (e.g., key constraints), can easily be specified in a form similar to (or derived from) SQL *create table* statements. In addition, integrity constraints (e.g., such as *end_time* being greater than *begin_time*) can be expressed. A tolerance is specified for each constraint. The tolerance is the amount of change or violation of the constraint that is acceptable. This is an important parameter since it can be used to tailor the quality of the watermark at the expense of greater change in the data. As mentioned earlier, if the tolerances are too low, it may not be possible to insert a watermark in the data. Various forms of expression are accommodated, e.g., in terms of arbitrary SQL queries over the relations, with associated requirements (usability metric functions). For example, the requirement that the result of the join (natural or otherwise) of two relations does not change by more than 3% can be specified.

Once usability metrics are defined and all other parameters are in place, the watermarking module (see Figure 10) initiates the process of watermarking. An undo/rollback log is kept for each atomic step performed (i.e., 1-bit encoding) until data usability is assessed and confirmed by querying the currently active usability plugins. This allows for rollbacks in the case when data quality is not preserved by the current atomic operation.

To validate this consumer driven design the authors perform a set of experiments showing how, for example, watermarking with classification preservation can be enforced through the usability metric plugin mechanisms. Moreover, the solution is proved experimentally on real data to be extremely resilient to random alterations and uninformed alteration attacks. This is due to its distribution-based encoding which can naturally survive such alterations. For example, alter-

```
det(attribute, wm_key, db_primary_key, subset_sz, v_false, v_true, c, map[], subset_boundaries[])
    sorted_attribute ← sort_on_normalized_crypto_hash(wm_key,db_primary_key,wm_key)
    read_pipe ← null
    do { tuple ← next_tuple(sorted_attribute) }
    until (exists idx such that (subset_boundaries[idx] == tuple))
    current_subset ← idx
    while (not(sorted_attribute.empty())) do
        do {
            tuple ← next_tuple(sorted_attribute)
            read_pipe = read_pipe.append(tuple)
        } until (exists idx such that (subset_boundaries[idx] == tuple))
        subset_bin ← (at most subset_sz elements from read_pipe, excluding last read)
        read_pipe.remove_all_remaining_elements_but_last_read()
        if (map[current_subset]) then
            mark_data[current_subset] ← decode (subset_bin, v_false, v_true, confidence)
            if (mark_data[current_subset] != DECODING_ERROR)
                    then map[current_subset] ← true
        current_subset ← idx
    return mark_data, map
```

Fig. 9. Watermark Detection (version using subset markers shown).

ing the *entire* watermarked data set within 1% of its original values only yields a distortion of less than 5% in the detected watermark.

The authors also propose a set of improvements and discuss several properties of the solutions.

- Embedding Optimizations: As the encoding resilience is dependent on a set of parameters (e.g., c, $subset\_size$, v_{false}, v_{true}), an automatic fine-tuning mechanism for searching a near-optimum in this parameter space is proposed. Additionally, the watermarking process could be trained to be resilient to a set of transformations expected from any potential attacker.
- Blind Watermarking: The method does not require the availability of the un-watermarked data at detection time.
- On-the-Fly Updatability: The authors also discuss mechanisms for handling dynamic data updates. Several scenarios of interest are: (i) updates that add fresh tuples to the already watermarked data set, (ii) updates that remove tuples from the already watermarked data and (iii) updates that alter existing tuples.

4 Categorical Types

So far we have explored the issue of watermarking *numeric* relational content. Another important relational data type to be considered is categorical data.

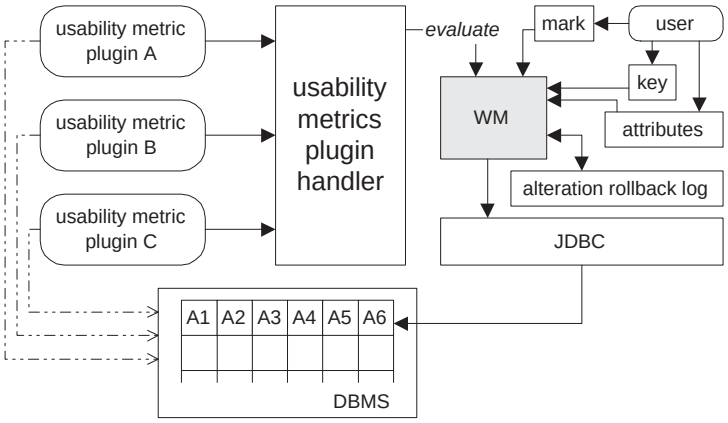

Fig. 10. Overview of the **wmdb.*** package.

Categorical data is data drawn from a discrete distribution, often with a finite domain. By definition, it is either non-ordered (nominal) such as gender or city, or ordered (ordinal) such as high, medium, or low temperatures. There are a multitude of applications that would benefit from a method of rights protection for such data. In this section we propose and analyze watermarking relational data with *categorical* types.

Additional challenges in this domain derive from the fact that one cannot rely on arbitrary small (e.g., numeric) alterations to the data in the embedding process. Any alteration has the potential to be significant, e.g., changing DEPARTURE_CITY from "Chicago" to "Bucharest" is likely to affect the data quality of the result more than a simple change in a numeric domain. There are no "epsilon" changes in this domain. This completely discrete characteristic of the data requires discovery of fundamentally new bandwidth channels and associated encoding algorithms.

4.1 The Adversary Revisited

We outlined above a set of generic attacks in a relational data framework. Here we discuss additional challenges associated with categorical data types.

A3. Alteration. In the categorical data framework, subset alteration is intuitively quite expensive from a data-value preservation perspective. One has also to take into account semantic consistency issues that become immediately visible because of the discrete nature of the data.

A6. Attribute Remapping. If data semantics allow it, re-mapping of relation attributes can amount to a powerful attack that should be carefully considered. In other words, if Mallory can find an even partial value-preserving mapping (the resulting mapped data set is still valuable for illicit purposes) from the original attribute data domain to a new domain, a watermark should hopefully survive

such a transformation. The difficulty of this challenge is increased by the fact that there likely are many transformations available for a specific data domain. This is thus a hard task for the generic case. One special case is primary key re-mapping.

4.2 A Solution

In [25], [36] Sion et. al. introduce a novel method of watermarking relational data with categorical types, based on a set of new encoding channels and algorithms. More specifically, two domain-specific watermark embedding channels are used, namely (i) *inter-attribute associations* and (ii) *value occurrence frequency-transforms* of values.

Overview. The solution starts with an initial user-level assessment step in which a set of attributes to be watermarked are selected. In its basic version, watermark encoding in the *inter-attribute association* channel is deployed for each attribute pair (K, A) in the considered attribute set. A subset of "fit" tuples is selected, as determined by the association between A and K. These tuples are then considered for mark encoding. Mark encoding alters the tuple's value according to secret criteria that induces a statistical bias in the distribution for that tuple's altered value. The detection process then relies on discovering this induced statistical bias.

The authors validate the solution both theoretically and experimentally on real data (Wal-Mart sales). They demonstrate resilience to both alteration and data loss attacks, for example being able to recover over 75% of the watermark from under 20% of the data.

Solution Details. For illustration purposes, let there be a set of discrete attributes $\{A, B\}$ and a primary data key K, not necessarily discrete. Any attribute $X \in \{A, B\}$ can yield a value out of n_X possibilities (e.g., city names, airline names). Let the number of tuples in the database be N. Let $b(x)$ be the number of bits required for the accurate representation of value x and $msb(x, b)$ its most significant b bits. If $b(x) < b$, x is left-padded with $(b - b(x))$ zeroes to form a b-bit result. Similarly, $lsb(x, b)$ is used to denote the least significant b bits of x. If by wm denotes a watermark to be embedded, of length $|wm|$, $wm[i]$ will then be the i-th bit of wm. Let $set\_bit(d, a, b)$ be a function that returns value d with the bit position a set to the truth-value of b. In any following mathematical expression let the symbol "&" signify a *bit-AND* operation. Let $T_j(X)$ be the value of attribute X in tuple j. Let $\{a_1, ..., a_{n_A}\}$ be the discrete potential values of attribute A. These are distinct and can be sorted (e.g., by ASCII value). Let $f_A(a_j)$ be the normalized (to 1.0) occurrence frequency of value a_j in attribute A. $f_A(a_j)$ models the de-facto occurrence probability of value a_j in attribute A.

The encoding algorithm (see Figure 11) starts by discovering a set of "fit" tuples determined directly by the association between A and the primary relation key K. These tuples are then considered for mark encoding.

Step One. A tuple T_i is said to be "fit" for encoding iff $H(T_i(K), k_1) \bmod e = 0$, where e is an adjustable encoding parameter determining the percentage of considered tuples and k_1 is a secret $max(b(N), b(A))$-bit key. In other words, a

```
wm_embed_alt(K,A,wm,k1,e,ECC)
  wm_data ← ECC.encode(wm, wm.len)
  idx ← 0
  for (j ← 1; j < N; j ← j + 1)
    if (H(Tj(K), k1) mod e = 0) then
      t ← set_bit(H(Tj(K), k1), 0, wm_data[idx])
      Tj(A) ← at
      embedding_map[Tj(K)] ← idx
      idx ← idx + 1
  return embedding_map
```

Fig. 11. Encoding Algorithm (alternative using embedding map shown)

tuple is considered "fit" if its primary key value satisfies a certain secret criteria (similar criteria are found in various frameworks, e.g., [16]). The fit tuples set will then contain roughly $\frac{N}{e}$ elements.

The "fitness" selection step provides several advantages. On the one hand this ensures secrecy and resilience and, on the other hand, it effectively "modulates" the watermark encoding process to the actual attribute-primary key association. Additionally, this is the place where the cryptographic safety of the hash one-wayness is leveraged to defeat invertibility attacks (**A5**). If the available embedding bandwidth $\frac{N}{e}$ is greater than the watermark bit-size $|wm|$, error correcting codes (ECC) are deployed that take as input a desired watermark wm and produce as output a string of bits wm_data of length $\frac{N}{e}$ containing a redundant encoding of the watermark, tolerating a certain amount of bit-loss, $wm\_data = ECC.encode(wm, \frac{N}{e})$.
Step Two. For each "fit" tuple T_i, we encode one bit by altering $T_i(A)$ to become $T_i(A) = a_t$ where

$$t = set\_bit(msb(H(T_i(K), k_1), b(n_A)), 0, wm\_data[msb(H(T_i(K), k_2), b(\frac{N}{e}))])$$

and k_2 is a secret key $k_2 \neq k_1$. In other words, a secret value of $b(n_A)$ bits is generated – depending on the primary key and k_1 – and then its least significant bit is forced to a value according to a corresponding position in wm_data (random, depending on the primary key and k_2). The new attribute value is thus selected by the secret key k_1, the associated relational primary key value and a corresponding bit from the watermark data wm_data.

In the decoding phase (see Figure 13), the first aim is to discover the embedded wm_data bit string. The same criteria for discovering "fit" tuples is used. For each "fit" tuple T_i, with $T_i(A) = a_t$, its corresponding entry in the result bit string is set to $(t\&1)$

$$wm\_data[msb(H(T_j(K), k_2), b(\frac{N}{e}))] \leftarrow (t\&1)$$

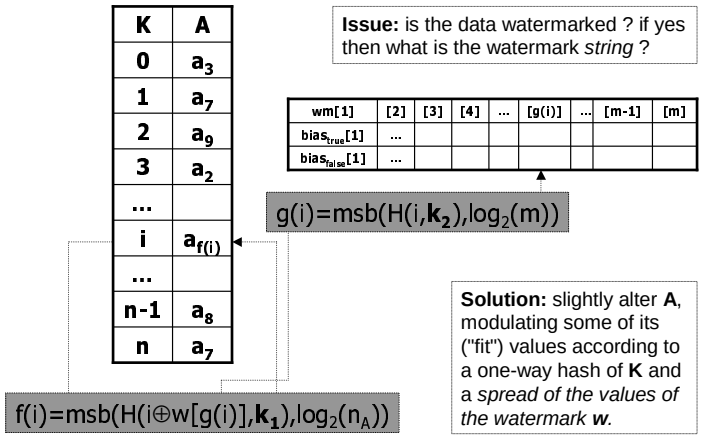

Fig. 12. Overview of multi-bit watermark encoding.

Once *wm_data* (possibly altered) is available, the error correcting mechanism is invoked to generate the "closest", most likely, corresponding watermark $wm = ECC.decode(wm\_data, |wm|)$.

The authors propose a natural extension to the above solution aimed at defeating vertical partitioning attacks (**A1.b**). Instead of relying on the association between the primary key and A, the extended algorithm considers *all* pairs of attributes and embeds a watermark separately in *each* of these associations. Additionally, if data constraints allow, the authors propose watermarking each and every attribute pair by first building a closure for the set of attribute pairs over the entire schema that minimizes the number of encoding interferences while maximizing the number of pairs watermarked. To solve the issue of interference, maintaining a mark "interference graph" is proposed.

The proposed extension features a particular issue of concern in certain cases of multi-attribute embeddings where two non-key attributes are used in the encoding, i.e., mark(A,B). Because of the correlation between the watermarking alteration (the newly selected value $T_i(B) = b_t$) and its actual location (determined by the fitness selection, $H(T_i(A), k_1)$ and e), sometimes Mallory can mount a special attack with the undesirable result of revealing some of the mark bit embedding locations. This occurs if the fitness criteria decides that a particular value of A yields a tuple fit and that value of A appears then in multiple (statistically significant number of) different tuples. This is possible only if A is not a primary key but rather another categorical attribute, with repeating duplicate values.

```
wm_dec_alt(K,A,k1,e,ECC,embed_map)
  for (j ← 1; j < N; j ← j + 1)
    if (H(Tj(K), msb(k, b(K))) mod e = 0) then
    determine t such that Tj(A) = at
    wm_data[embed_map[Tj(K)]] = t&1
  wm ← ECC.decode(wm_data, wm.length)
  return wm
```

Fig. 13. Decoding Algorithm (alternative using embedding map shown)

The authors propose a set of solutions to this issue, including composing the actual watermark encoding out of a combination of several different sub-encodings, each in turn using a different k_1 value. Each such sub-encoding will ignore all tuples with previously seen values of the attribute A (in the fitness criteria). While each of these "low impact" encodings would be weaker than the original solution, their combined "sum" can be made arbitrarily strong, by increasing their number. At the same time correlation attacks would be defeated, as each of the encodings would use a different key thus making such attacks impossible "across" the encodings.

The authors further discuss additional extensions and properties of the solution, including the following.

- Consumer-Driven Design: The solution features a consumer-driven design. Each property of the database that needs to be preserved is written as a constraint on the allowable change to the dataset. The watermarking algorithm is then applied with these constraints as input and re-evaluates them continuously for each alteration. A backtrack log is kept to allow undo operations in case certain constraints are violated by the current watermarking step.
- Incremental Updatability: The solution supports incremental updates naturally. As updates occur to the data, the resulting tuples can be evaluated on the fly for "fitness" and watermarked accordingly.
- Blind Watermarking: The method does not require the availability of the un-watermarked data at detection time.
- Minimizing Alteration Distance: An interesting problem to consider is the case when, for a given "fit" tuple, certain alterations would be preferred to others (e.g., changing "Chicago, O'Hare" into "Chicago" is preferred to "Las Vegas"). The authors propose to handle this scenario by a modified encoding procedure that naturally accommodates and minimizes such an "alteration distance" metric.
- Extreme Vertical Partitioning: To counter extreme vertical partitioning attacks in which only a single attribute A is preserved in the result, the authors

propose to encode a watermark in one of the only remaining characteristic properties, namely the value occurrence frequency distribution for each possible value of A. To do so a scheme of watermarking for numeric sets [30] can be applied in this "frequency" domain.
- Multi-Layer Self-Reinforcing Watermarks: To counter the scenario where Mallory gains knowledge, e.g., during a court hearing, of a multiply-used encoding key, the authors propose to embed multiple (i) weak watermarks with different secret keys and reveal in court only a certain subset of these, or (ii) *self-re-enforcing* pairs of watermarks $(w_1, w_2)_i$ with different keys $(k_1^1, k_2^1, k_1^2, k_2^2)_i$ such that, for example, altering w_2 will result in enforcing w_1.
- Multiple Data Sources: The paper also points out that the solution handles recovering watermarks from data derived from multiple data sources. This scenario is of particular interest for example in the case of an equiJOIN performed between two data sets. Because watermarks rely on a bias in the association between attributes, they can be naturally retrieved from such JOIN result under certain reasonable assumptions.
- Categorical and Numerical Data Types: Watermarking at the intersection of categorical and numerical types is also explored. It is of interest to provide a rights assessment mechanism that could not only prove rights but also that the associated data sets were actually produced "together"; this is relevant for example if the intrinsic value of the data lies in the actual *combination* of the two data types. The authors introduce initial ideas.
- Bijective Attribute Re-mapping: To handle a scenario in which categorical attributes are re-mapped through a bijective function to a new data domain, the authors propose to discover the inverse mapping. This is possible if the initial data domain features distinguishing properties (e.g., value occurrence frequency histogram) that are likely to be preserved in the mapped result.

5 Related Work

So far we have discussed a set of relational data types and associated watermarking methods enabling future rights assessment proofs. We now survey a number of related research efforts that explore Information Hiding and Watermarking for relational data in other security contexts such as privacy enforcement and license violators tracing.

5.1 Privacy and Rights Protection

In [4] Bertino et. al. explore issues at the intersection of two important dimensions in data-centric assurance, namely rights assessment and privacy, in the broader context of medical data. A unified framework is introduced that combines binning and watermarking techniques for the purpose of achieving both data privacy and the ability to assert rights.

The system design borrows components from existing work. More specifically, the binning method (for k-anonymity) is built upon an earlier approach

of generalization and suppression by allowing a broader concept of generalization. Similar to the *consumer-driven* paradigm discussed earlier in this chapter, to ensure data usefulness, binning is constrained by usage metrics that define maximal allowable information loss. An initial binning stage is followed then by watermarking. The framework then deploys a version of the encoding for categorical types [36] by Sion et. al. in a hierarchical fashion, for the purpose of defeating a data generalization attack of concern in this framework. The paper then explores whether watermarking can adversely interfere with binning and conclude that the interaction is safe. Experiments were conducted aimed at validating the robustness of the proposed framework.

5.2 Fingerprinting

Another example application of Resilient Information Hiding as a tool aiding rights management, is its deployment to "track" license violators by hiding a specific mark inside the Work, uniquely identifying the party it was sold/outsourced to. This application is commonly referred to as *fingerprinting*. If the Work would then be found in the public domain, that mark could be used to assess the source of the leak.

One significant matter of concern in fingerprinting are collusion attacks. In a collusion attack, multiple attackers "collude" by obtaining multiple copies of the same Work (e.g., by purchasing it separately under different identities) watermarked with different marks, in the hope of "combining" the different copies into a single un-watermarked version. Defending against this attack is not possible in the general case when the number of colluding partners cannot be upper bounded. If this upper bound can be determined however, several results provide appropriate coding techniques that allow tracing even in the case of collusion under minimal assumptions [5] [6] [13].

For relational data, the issue of fingerprinting has been discussed by Li et. al. in [21] where they propose to deploy their multi-bit watermarking method [19] for this very purpose. To handle collusion attacks the authors defer to research in [5] [6] [13]. This work is discussed in more detail elsewhere in this book.

5.3 Tamper Detection through Fragile Watermarking

In [17] Li et. al. explore the issue of detecting malicious alterations to data by embedding a "fragile" watermark in the data. While in this chapter we presented watermarking as a technique deploying Information Hiding for the purpose of rights assessment, in this context, "watermark" is attached to a different semantics. Whereas in rights assessment, a watermark features resilience to value-preserving data alterations, for the purposes of tamper detection, the "watermark" will be "fragile" so as to become a detector for exactly such alterations. The authors also propose to allow this watermark to point at the locations where alterations have occurred in the data.

At an overview level, the method proceeds as follows. The data is partitioned into secret subsets; a keyed cryptographic hash of each such subset (in effect the

traditional message authentication code MAC) is then embedded in the group by re-ordering its items with respect to a canonical ordering, based on a cryptographic hash of their primary key attribute. The encoding is claimed fragile enough to be impacted by even minor alterations to the data with reasonable probabilities. Additionally, the encoding can pinpoint at the exact location of the alteration with the granularity of a subset.

Compared with traditional authentication techniques (e.g., appending signatures of MACs) such a technique can become of relevance, e.g., when the overhead of storing and managing the signatures or MACs for a large number of entities is not negligible. This is why it is important to further explore and understand fragile watermarking scenarios. This work is discussed in more detail elsewhere in this book.

5.4 Query Learnability and Consumer-Driven Watermarking

In [12] Gross-Amblard introduce interesting theoretical results investigating alterations to relational data (or associated XML) in a consumer-driven framework in which a set of parametric queries are to be preserved up to an acceptable level of distortion.

The author first shows that the main difficulty preserving such queries "is linked to the informational complexity of sets defined by queries, rather than their computational complexity" [12]. Roughly speaking, if the family of sets defined by the queries is not *learnable* [37], no query-preserving data alteration scheme can be designed.

In a second result, the author shows that under certain assumptions (i.e., query sets defined by first-order logic and monadic second order logic on restricted classes of structures – with a bounded degree for the Gaifman graph or the tree-width of the structure) a query-preserving data alteration scheme exists.

This research is important as it has the potential to enable a better understanding of consumer-driven watermarking designs. For example, as database instances are often having a bounded degree Gaifman graph (or a bounded tree-width), these can now be measured and the information capacity of a query-preserving alteration channel can be computed. This is of interest in the case of extremely restrictive constraints, e.g., when it is not clear if watermarking can yield enough resilience.

6 State of The Art and the Future

Watermarking in relational frameworks is a relatively young technology that has begun its maturity cycle towards full deployment in industry-level applications. Many of the solutions discussed above have been prototyped and validated on real data. Patents have been filed for several of them, including Agrawal et.al. [1, 16] and Sion et.al. [29, 30, 32, 33] [34] [25, 27, 36]. In the next few years we expect these solutions to become available commercially, tightly integrated

within existing DBMS (e.g., DB2 [10]) or as stand-alone packages that can be deployed simultaneously on top of multiple data types and sources. Ultimately, we believe the process of resilient information hiding will become available as a secure mechanism for not only rights protection but also data tracing and authentication in a multitude of discrete data frameworks.

7 Conclusions

In this chapter we explored how Information Hiding can be successfully deployed as a tool for Rights Assessment for discrete digital Works. We analyzed solutions for resilient Information Hiding for relational data, including numeric and categorical types.

A multitude of associated future research avenues present themselves in a relational framework, including: the design of alternative primary or pseudo-primary key independent encoding methods, a deeper theoretical understanding of limits of watermarking for a broader class of algorithms, the ability to better defeat additive watermark attacks, an exploration of zero-knowledge watermarking etc.

Moreover, while the concept of on-the-fly quality assessment for a consumer-driven design has the potential to function well, another interesting avenue for further research would be to augment the encoding method with direct awareness of semantic consistency (e.g., classification and association rules). This would likely result in an increase in available encoding bandwidth, thus in a higher encoding resilience. One idea would be to define a generic language (possibly subset of SQL) able to naturally express such constraints and their propagation at embedding time.

Additionally, of particular interest for future research exploration, we envision cross-domain applications of Information Hiding in distributed environments such as sensor networks, with applications ranging from resilient content annotation to runtime authentication and data integrity proofs.

8 Acknowledgments

We would like to thank our reviewers for excellent feedback and suggestions.

References

1. Rakesh Agrawal, Peter J. Haas, and Jerry Kiernan. Watermarking relational data: framework, algorithms and analysis. *The VLDB Journal*, 12(2):157–169, 2003.
2. Michael Arnold, Stephen D. Wolthusen, and Martin Schmucker. *Techniques and Applications of Digital Watermarking and Content Protection*. Artech House Publishers, 2003.
3. Mauro Barni and Franco Bartolini. *Watermarking Systems Engineering: Enabling Digital Assets Security and Other Applications*. Marcel Dekker, 2004.

4. Elisa Bertino, Beng Chin Ooi, Yanjiang Yang, and Robert H. Deng. Privacy and ownership preserving of outsourced medical data. In *Proceedings of the International Conference on Data Engineering*, pages 521–532, 2005.
5. D. Boneh and J. Shaw. Collusion-secure fingerprinting for digital data. *Lecture Notes in Computer Science*, 963:452–464, 1995.
6. D. Boneh and J. Shaw. Collusion-secure fingerprinting for digital data. *IEEE Transactions on Information Theory*, 44(5):1897–1905, 1998.
7. E.F. Codd. A Relational Model of Data for Large Shared Data Banks. *Communications of the ACM*, 13(6):377–387, 1970.
8. I. Cox, J. Bloom, and M. Miller. Digital watermarking. In *Digital Watermarking*. Morgan Kaufmann, 2001.
9. Scott Craver, Nasir Memon, Boon-Lock Yeo, and Minerva M. Yeung. Resolving rightful ownerships with invisible watermarking techniques: Limitations, attacks, and implications. *IEEE Journal of Selected Areas in Communications*, 16(4):573–586, 1998.
10. The IBM DB2 Universal Database. Online at `http://www.ibm.com/software/data/db2`.
11. Joachim Eggers and Bernd Girod. *Informed Watermarking*. Kluwer Academic Publishers, 2002.
12. David Gross-Amblard. Query-preserving watermarking of relational databases and xml documents. In *Proceedings of the Nineteenth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 191–201, New York, NY, USA, 2003. ACM Press.
13. H. Guth and B. Pfitzman. Error and collusion secure fingerprinting for digital data. In *Proceedings of the Information Hiding Workshop*, 1999.
14. Neil F. Johnson, Zoran Duric, and Sushil Jajodia. *Information Hiding: Steganography and Watermarking - Attacks and Countermeasures*. Kluwer Academic Publishers, 2001.
15. S. Katzenbeisser and F. Petitcolas (editors). *Information Hiding Techniques for Steganography and Digital Watermarking*. Artech House, 2001.
16. J. Kiernan and R. Agrawal. Watermarking relational databases. In *Proceedings of the 28th International Conference on Very Large Databases VLDB*, 2002.
17. Yingjiu Li, Huiping Guo, and Sushil Jajodia. Tamper detection and localization for categorical data using fragile watermarks. In *DRM '04: Proceedings of the 4th ACM workshop on Digital rights management*, pages 73–82, New York, NY, USA, 2004. ACM Press.
18. Yingjiu Li, Vipin Swarup, and Sushil Jajodia. Constructing a virtual primary key for fingerprinting relational data. In *DRM '03: Proceedings of the 2003 ACM workshop on Digital rights management*, pages 133–141, New York, NY, USA, 2003. ACM Press.
19. Yingjiu Li, Vipin Swarup, and Sushil Jajodia. A robust watermarking scheme for relational data. In *Proceedings of the Workshop on Information Technology and Systems (WITS)*, pages 195–200, 2003.
20. Yingjiu Li, Vipin Swarup, and Sushil Jajodia. Defending against additive attacks with maximal errors in watermarking relational databases. In *Proceedings of the IFIP WG 11.3 Working Conference on Data and Application Security*, pages 81–94, 2004.
21. Yingjiu Li, Vipin Swarup, and Sushil Jajodia. Fingerprinting relational databases: Schemes and specialties. *IEEE Transactions on Dependable and Secure Computing*, 2(1):34–45, 2005.

22. Chun-Shien Lu. *Multimedia Security: Steganography and Digital Watermarking Techniques for Protection of Intellectual Property*. Idea Group Publishing, 2004.
23. Bruce Schneier. *Applied Cryptography: Protocols, Algorithms and Source Code in C*. Wiley & Sons, 1996.
24. Husrev T. Sencar, Mahalingam Ramkumar, and Ali N. Akansu. *Data Hiding Fundamentals And Applications: Content Security in Digital Multimedia*. ELSEVIER science and technology books, 2004.
25. Radu Sion. Proving ownership over categorical data. In *Proceedings of the IEEE International Conference on Data Engineering ICDE*, 2004.
26. Radu Sion. *Rights Assessment for Discrete Digital Data, Ph.D. dissertation*. Computer Sciences, Purdue University, 2004.
27. Radu Sion. wmdb.*: A suite for database watermarking (demo). In *Proceedings of the IEEE International Conference on Data Engineering ICDE*, 2004.
28. Radu Sion and Mikhail Atallah. Attacking digital watermarks. In *Proceedings of the Symposium on Electronic Imaging SPIE*, 2004.
29. Radu Sion, Mikhail Atallah, and Sunil Prabhakar. On watermarking numeric sets. Online at `https://www.cerias.purdue.edu/tools_and_resources/bibtex_archive/`, 2001.
30. Radu Sion, Mikhail Atallah, and Sunil Prabhakar. On watermarking numeric sets. In *Proceedings of IWDW 2002, Lecture Notes in Computer Science*. Springer-Verlag, 2002.
31. Radu Sion, Mikhail Atallah, and Sunil Prabhakar. Power: Metrics for evaluating watermarking algorithms. In *Proceedings of IEEE ITCC 2002*. IEEE Computer Society Press, 2002.
32. Radu Sion, Mikhail Atallah, and Sunil Prabhakar. Watermarking databases. Online at `https://www.cerias.purdue.edu/tools_and_resources/bibtex_archive/`, 2002.
33. Radu Sion, Mikhail Atallah, and Sunil Prabhakar. Rights protection for relational data. In *Proceedings of the ACM Special Interest Group on Management of Data Conference SIGMOD*, 2003.
34. Radu Sion, Mikhail Atallah, and Sunil Prabhakar. Relational data rights protection through watermarking. *IEEE Transactions on Knowledge and Data Engineering TKDE*, 16(6), June 2004.
35. Radu Sion, Mikhail Atallah, and Sunil Prabhakar. Resilient rights protection for sensor streams. In *Proceedings of the Very Large Databases Conference VLDB*, 2004.
36. Radu Sion, Mikhail Atallah, and Sunil Prabhakar. Ownership proofs for categorical data. *IEEE Transactions on Knowledge and Data Engineering TKDE*, 2005.
37. L. G. Valiant. A Theory of the Learnable. In *Proceedings of the Symposium on the Theory of Computing*, pages 436–445, 1984.